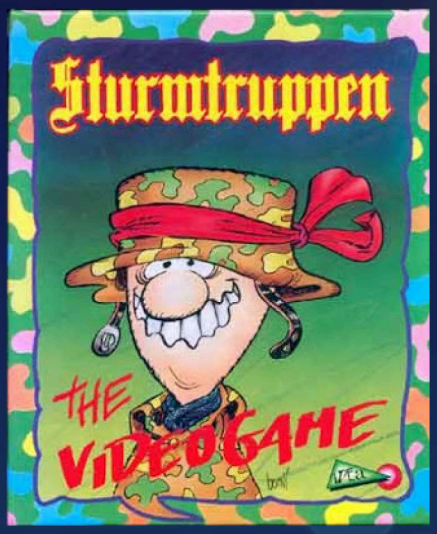
RetroMagazine World

future days are back

Issue 2 Year 1 - August 2020 - https://www.retromagazine.net - Free Publishing



Game review: Sturmtruppen (Amiga)

Back to the past with Windows 2000

Game review: Cyborg Z (MSX)

Game & Watch: interview with Luca Antignano "MADrigal's simulators"

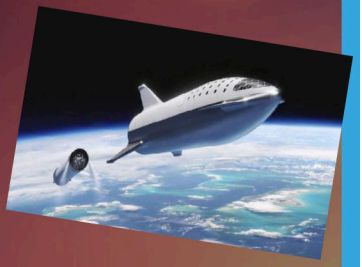
Game review: Moon Cresta (Arcade)

Game review: Wonderboy The Dragon's Trap (1989 & remake)

Preview of Ellenica (Megadrive)

Game review: Raging Beast (C64)

- LM80C: a 2019 Z80-based self-built microcomputer - part 1
- Abbreviations & shortcuts on using a Graphic Interface
- The AREXX language for Amiga - part 1
- How to make a splash screen for Amstrad CPC in SCR format
- The Olivetti M20 and the history of a web site
- The 1st RMW 8-bit Home Computer Chess Tournament



Holiday time, memory time...

Summer, with its torrid heat and hot nights came to visit all of us again. Probably now more than ever warmth and temperatures above average have been expected with such trepidation. After a horrible winter and spring, this summer is not only synonymous with holidays, but also a slow return to the normal life for some of us.

Who's writing have been living abroad for few years now and summer is one of the most awaited moments to be able to return to Italy and embrace friends and family. This year you can easily imagine how ardently I was waiting for the possibility to travel again and return to the places of my youth. For those who live far from their home country the chance to return once or twice a year is like browsing through a memory album. Finding places and people you haven't seen in a long time makes you want to know what happened in the meantime and likewise the possibility of making a comparison with what it was and what it is... The funniest thing, at least for me, is noticing how the little changes are much more impressive than the big ones. They're not so obvious, and when you see them, you make an immediate comparison.

Some of you may wonder what the link of this editorial is to our common passion. Easily said! Because of the complexity of the move and the chronic lack of space in our homes (at least in mine...) I had to give up transferring a substantial part of my retro material. Books, magazines, notebooks, remained in my house in Italy. So, coming back and finding this stuff is like opening a memory chest. It is true that, as I have also written in other editorials, we can find retro magazines and books on the Internet, but these are my personal ones. Reading their pages, immediately arise the memories, the doubts, the things that were not clear to me at the time and that I now easily understand... Bringing me back in time.

Strangely, I feel a mixture of feelings; I would like to take this material with me, but at the same time I like the idea of coming back once in while and finding them for a real return to the past.

What about you? Do you have your own treasure chest?
If you want to tell us your stories, please contact us through the references below.

The number you have downloaded is full of interesting and hopefully stimulating content. Before leaving you to enjoy it, on behalf of the entire RMW Team, I would like to wish you a very nice summer!

Francesco Fiorentini

Email: retromagazine.redazione@gmail.com
Facebook: www.facebook.com/RetroMagazine-2005584959715273/
Twitter: twitter.com/RetroMagazineW

TABLE OF CONTENTS

People involved in the preparation of this issue of RetroMagazine World:

- Alberto Apostolo
- Carlo N. Del Mar Pirazzini
- Daniele Brahimi
- Davide Bucci
- David La Monaca
- Ermanno Betori
- Flavio Soldani
- Francesco Fiorentini
- Gianluca Girelli
- Giorgio Balestrieri
- Leonardo Miliani
- Marco Fiaschi
- Marco Pistorio
- Mathias Lorenz
- Michele Ugolini
- Querino Ialongo
- Sakis Kaffesakis

- Graphics by Irene Valeri
- Cover by Flavio Soldani
- Proof-reading by
  Francesco Fiorentini
  David La Monaca
  Robin Jubber
  Ralf Quint
  Carlo NDM Pirazzini
  Gianluca Girelli

The Olivetti M20 and the history of a website

by Davide Bucci

1 - Introduction

The Olivetti M20 was a remarkable computer, developed at a time when it was not yet clear that Intel processors and MS-DOS would play a big role in the years to come. Ultimately, it would not have direct successors and did not create an ecosystem large enough to survive more than a few years. Nonetheless it found a small niche, notably in Europe and contributed to the daily work of many small businesses.

This article presents the M20, as well as the operating system that Olivetti developed for it, the PCOS. I have been and still am the webmaster of a website, dedicated to this machine, which went online in 2005, so the second part of this article is more of a personal nature and describes the history of my computer as well as that of the website I dedicated to it.

Figure 1 - The M20 case, designed by Ettore Sottsass

2 - The beat of a different drum, the Z8001

Development of the M20 (internally called PC1000) started in 1979 in the Olivetti Advanced Technology Centre (ATC) in Cupertino, California, not far from the Apple headquarters. Olivetti presented the computer to the public on March 31, 1982, in the magnificent castle of Agliè, not too far from Ivrea in Italy [1]. They had already used the M20 name for a older model of typewriter in 1920 and hoped to repeat the success for the then blossoming personal computer market. The newborn was based on the Zilog Z8001 processor, like all machines of the 'LINEA 1' (L1) family, completed by the M30, M40 and M60 minicomputers. The M20 ran a proprietary operating system developed specifically for it, the Professional Computer Operating system or PCOS-8000. One colourful detail is that several Italian ministers and political representatives were invited to the Agliè meeting. Presenting a new computer in Italy in this way was not only a minor technological event, but more a political and (why not?) a cultural event, however the computer was actually distributed worldwide [2-4] and attracted a certain level of interest. The title of this paragraph paraphrases the subtitle of an American review [3].

Olivetti was well known for their industrial design and the M20 was no exception. A modern-looking plastic case was designed by Ettore Sottsass (1917-2007) and was praised at the time. The designer had a long history of collaboration with Olivetti, notably, he was in charge of the industrial design of the Elea 9003 in 1959, one of the first fully transistorised computers in the world. In 1969, he also drew the stylish Valentine typewriter, attractively adorned in a garish red colour. The M20 hosted a respectably sized motherboard, a well equipped keyboard (not detachable) as well as two 5.25" floppy disk drives or one floppy disk drive and one hard disk. A separate 12" monitor could be put on top of the case and oriented. The result can be seen in figure 1 and still retains a certain retro-futuristic appeal to the modern eye.

3 - The hardware

The motherboard, shown in figure 2, occupied practically all the base of the computer, hence the choice of an integrated keyboard was dictated by technical constraints. The motherboard contained a 4 MHz Z8001 processor,

using a real 16 bit bus, 128 KB of RAM, a printer interface and an RS232 serial interface with a maximum speed of 9600 baud. The RAM could be expanded up to a maximum of 512 KB by using up to three expansion cards. Two slots were available for expansions such as an IEEE interface card, hard disk controller and so on. The build quality was excellent and the system was modular. Assembly and disassembly was a matter of a few screws and some clever latches.

The keyboard was slightly noisy, but fast and excellent for touch typing. Italian readers may notice in figure 3 that it conserved the classic QZERTY layout for typewriters, as the modern Italian QWERTY layout plus the accented letters was yet to come, introduced by the same Olivetti with the M24. Olivetti clearly wanted people to type as comfortably as possible and aimed to people used to typewriters. Contemporary reviews [3] tend to explicitly mention the absence of separate function keys, however, special key functions were accessible via two coloured keys named Command and CTRL, these were pressed and held down before pressing another key. A detachable legend could be inserted in a slot on top of the keyboard, to indicate the function of each key. Two additional keys, called S1 and S2, could be assigned various functions. The most evident omission is the backspace key, normally obtained with CTRL+H (tab is obtained with CTRL+I). The S2 key is exactly at the place where one would expect a backspace key, but is configured by default as a Carriage-Return. Any key could be reprogrammed, for example, if you thought you could not live without a backspace key, you could type 'CK &C3, 8' to assign the S2 key (scan code C3 in hexadecimal) to that function (ASCII code 8 = CTRL+H). You can even make this permanent, with the PSAVE command. A small beeper mounted on the keyboard PCB was the only sound device available.

Olivetti was very attentive to international markets and many tailored versions of the M20 were made available. This caused some issues as the character sets differ, also some characters must be adapted if you type a BASIC source from a manual in English (a notable example is # of the English set that becomes £ in French or Italian). The issue was noticed at the time. It was mentioned in a French review [4] that a nationalised machine came with English manuals as the French ones were not yet ready. This is still true today as many of the manuals available are only in English [5].

Most machines came with a double 5.25" floppy disk drive unit whose 320 KB nominal disk capacity became 272 KB when a disk was formatted (called a "volume" in the PCOS jargon). Noticeable is the different formatting of the head 0/track 0 (MF, 128 bytes/sector) with respect

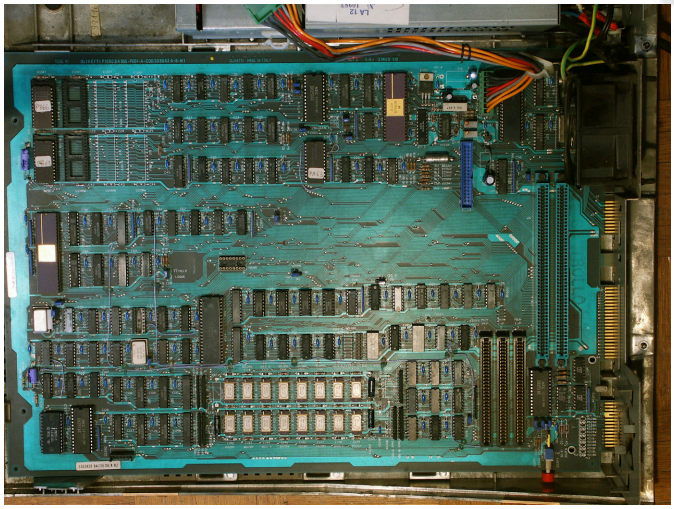


Figure 2 - The wide M20 motherboard

to the other tracks (MFM, 256 bytes/sector). Different configurations exist, an 11.5 MB Winchester drive (similar to the MFM ST251) was available to replace one of the floppy drives and 160 KB and 640 KB disk drives were also available. Unusually, in the standard hardware configuration, the M20 identifies the two floppy drives as 1: on the left and 0: on the right.

A range of printers was available from Olivetti. The PR1450S, PR1450G (G stands for "graphic"), PR1471, dot matrix and the PR2400 thermal. The M20 sports a standard Centronics parallel interface but, as for the RS232 interface and the screen, it employs a proprietary edge connector. An output mains plug from the computer allowed it to automatically power up the printer when the computer was switched on.

The M20 does not make a distinction between graphic modes and text modes, everything is drawn in a graphic screen with a 512x256 pixel resolution. This offered a great flexibility at the expense of a slight delay in some operations, such as text scrolling. The operating system uses either a 64x16 character grid or a more conventional 80x25 arrangement. The basic model came with a black and white display but an expensive (and therefore quite rare) option was a colour monitor that required special RAM expansion boards equipped with parallel to serial converters. These acted as a sort of a simple DMA circuitry able to direct the R G and B bit planes directly to the screen output.

The screen is powered by the computer with a 12V line and uses a single cable for the power supply and the video signals. It is nonstandard, signals are R, G, B, B/W, plus V and H sync (all TTL levels). The vertical sync operates at 68.2 Hz the horizontal at 18.7573 kHz. Those nonstandard frequencies make it difficult to find a compatible monitor if the original one is malfunctioning or is absent.

4 - The operating system

The PCOS operating system seems peculiar and obscure to the modern eye, but was not underpowered with respect to the DOS 1.0 that equipped the first IBM PC 5150 (presented to the press only 7 months earlier than the M20). It did not feature directories and subdirectories, but you could control file access with a password, even if the protection was not considered very strong. You could use abbreviations for commands such as 'VF' for 'VFORMAT' to format a new disk. It is worth mentioning that this command could fail to format a floppy disk that had already been formatted for MS-DOS. A radical but effective solution was to destroy all previous data with a strong magnet. The operating system was highly configurable, allowing you to decide which commands to keep resident, depending on the memory configuration and your convenience. Alternative operating systems were made available by Olivetti. In October 1982, Olivetti proposed an expansion board called Alternate Processor Board (APB), containing an Intel 8086 that allowed the M20 to run MS-DOS. Worthy of note was CP/M-86, an extension of the well-known CP/M for the Intel processor.

The Olivetti M20 uses standard DS-DD floppy disks and access was reasonably fast, a full disk format on a 320 KB unit takes 1 min and 18 s and a complete disk copy ('VCOPY' command) is done in 1 min 24 s. Booting PCOS 4.1a takes 27 s on my 512 KB machine. In general, the M20 was widely acknowledged to be a relatively powerful machine, the Z8001 was a decent processor for the time and was the computer was sold by Olivetti at an attractive price. A version of the Microsoft BASIC was shipped with the machine and execution speed of programs on the M20 was competitive with the original IBM 5150.

The successor to the M20 was called M24, it came out in 1983, but was based on the 8086 processor and MS-DOS. It was an incredibly successful and effective machine. Olivetti offered an "Alternate Processor Board", with a Z8001 for it, which allowed the M24 to run PCOS. A few years later, it became clear nonetheless that the death knell was definitely sounding for that operating system. Version 4.1a, developed in 1984, is the last version of the PCOS I am aware of.

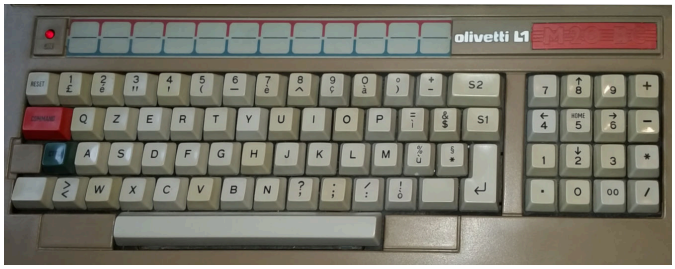


Figure 3 - Italian keyboard with the QZERTY layout

5 - My M20 and the birth of a website

The spring of 1993 was a sunny one in my small town, near Turin, in Italy. During one of the seemingly interminable afternoons that a teenager has on their hands, I went to visit a friend after school. Will was a very intelligent guy and an incredibly skilled guitarist for a boy who only just turned 14. Most of the time he played and I tried to learn something from him. That day he greeted me with a smile and said "Dave, I've got a computer that may interest you." showing me a brown and beige square shaped thing that was lying inert in his room. "My father got it from the car part shop nearby, they wanted to trash it as it's old. We can't use it: if you want, take it." Needless to say, I accepted.

"What is this... thing?" My father certainly wasn't enthusiast when he came that evening to pick me up with his car and saw me with that strange computer. We already had a 286 PC, but I saw some pictures on the twelve years old computer magazines I used to read at the time. That was, however, the first time I saw an Olivetti M20 in real life and I already liked much its strangely angular yet sleek shape.

I soon discovered my "new" computer was, indeed, useless without an appropriate operating system. It powered up, but it only had two disk drives and no hard disk. Once the self test terminated it remained utterly inert, waiting. After a few weeks, I asked Ugo, one of our neighbours at the time, for help. He used to live in the house next door to where I lived with my parents and he taught me basically everything I knew about computers. I had already received from him those old computer magazines I used to read with such delight. Ugo came to my rescue with his usual patience and kindness, offering me some old floppy disks for this strange machine. It turned out he had worked on a similar one ten years earlier.

I then learned that the M20 used a strange operating system called PCOS and I could play a little with the Microsoft Basic that came with it. I already knew this dialect, so I spent some hours playing and drawing some lines and circles on the screen. I also got some simple games (such as the one in figure 4) that were tiny yet charming.

That was fun, but without any manual and detailed information, there was not much I could do at the time. I kept my M20 in good shape, but not frequently used, until 2004. By then I was already living in France and, as I was a regular Internet user, I thought it would be

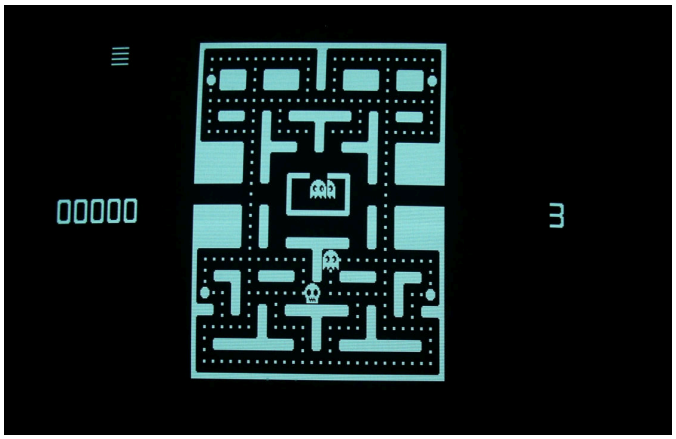


Figure 4 - A clone of Pac-Man

interesting to search for information about my old and strange computer. I discovered, with delight, that scans of some manuals were already available, but that information was scattered in different websites and was fragmented.

I decided to do something and I choose a solution that was obvious in an epoch of the Internet where social networks were yet to become widespread. I put together a web site, writing all the HTML and a bit of PHP myself. It went online in 2005 and still exists today [6], with the same old-fashioned look I chose back then. By the way, I would like to thank in particular Roberto Bazzano, who has hosted the site on his servers since 2007. Thanks to the website, I came in touch with many interesting and incredibly skilled people. The Olivetti M20 has a small, yet devoted, international community.

Around 2006, having become an electronics engineer, I tried improving my machine. It needed some repairs. The original power supply was not up to the task anymore and, after more than 20 years, the disk drives needed to be aligned. I managed to get a former Olivetti technician on the phone, who explained me the factory procedure to bring them back to life. I also studied the schematics of memory expansions and built one with two salvaged SIMM modules, bringing my machine to a whopping 512KB of RAM. Olivetti must have used a primitive CAD system at the time, their schematics were incredibly hard to read and literally full of errors. I suspect those used for the production were not the ones that appear in the hardware manual.

In all those years, I tried helping other people, trying to learn new things and documenting what I did with the machine, however the website hosts contributions from many other people, kind enough to send me a description of their experiences. This paper would not have been possible without the help I received from them via the website.

6 - Conclusion

In this article, I described a rather peculiar computer, the Olivetti M20. Very competitive with respect to the 5150 IBM PC, it was ultimately penalised by lack of software and perhaps general recognition. I described the hardware as well as the operating system, PCOS-8000. In the last part, I told the story of my machine (depicted in the figures) and how I decided to build a website dedicated to it. That allowed me to get in touch with many people from different countries who are passionate about the Olivetti M20.

One last note: I wrote the first version of this paper on my Olivetti M20 running Oliword 1.2, transferring the files with a RS232 link and using a simple Bash script to strip the control codes. Final editing was done on a modern MacBook Pro.

Acknowledgments

I would like to warmly thank my friend William Barbero, who more that 25 years ago allowed me to rescue my M20. Many thanks too to Ugo Garombo, who rescued me with the floppy disks and taught me what an operating system was. I also would like to thank Roberto Bazzano who kindly hosted the M20 website for many years.

This paper would have been probably awkward to read without the kind and attentive proofread by Chris Carter. The remaining errors are mine.

Bibliography

[1] M. Marinacci "Anteprima dell'Olivetti M20," MCmicrocomputer n°8, April 1982
[2] W. Marett "Italian micro enters New Zealand business market," Bits and Bytes n°9, June 1983
[3] S. Mello-Grand "Review of the Olivetti M20," Byte vol. 8 n°6 June 1983
[4] X. Bonfils, A. Pinaud, J.-P. Brunerie, "Au banc d'essai Olivetti M20," L'ordinateur individuel, n° 41, September 1982
[5] See for example: http://www.z80ne.com/m20/index.php?argument=sections/manuals/manuals.inc
[6] http://www.z80ne.com/m20

The LM80C Color Computer

A 2019 self-built Z80-based home computer - part 1

by Leonardo Miliani

With this article I would like to introduce you to a new computer, built in 2019: it is an innovative system, based on an 8-bit CPU and with 80 KB of total memory! This is the LM80C Color Computer, which I personally assembled. Yeah, it's a self-built computer.
But it's not a self-built computer that communicates with the serial via another computer or uses small LCD displays. This computer has its own keyboard, connects to a common home TV and operates independently: you can enter a program in BASIC into his memory and run it immediately, as you probably used to do when you were a kid with your own home computer.

Does it still make sense to propose in 2019 a self-built computer based on the Z80, which is an 8-bit CPU developed by Zilog in 1976? Let's say yes, if you are passionate about retrocomputing and/or nostalgic for 1980s systems. And it makes even more sense because it is “self-made”: you have to consider the pleasure of doing it working day by day, and, why not?, the enjoyment of playing games on a computer that’s entirely made by yourself.

I managed to take this pleasure away from me, creating a perfectly functioning 8-bit computer, complete with everything: an operating system with an integrated BASIC interpreter, a keyboard for input, the image on a screen, the sound output, the ability to interface with a remote computer via serial port. In the end, it is a perfectly usable system. You can create programs of any kind on it: I currently play Checkers, Reversi/Othello and Lunar Lander. In figure 1 you can see the system prototype on a breadboard:

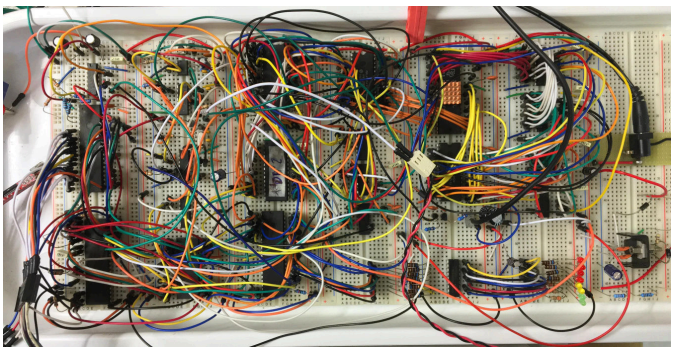


Figure 1: the LM80C prototype on the breadboard

How did I do that? With so much desire to do, stubbornness in keeping to move forward despite the many obstacles that I have had to face, commitment to studying a lot of technical documents about many integrated circuits, having a knack for programming (because the operating system is all written in Assembly). Moreover, the project is completely open-source: those who want to replicate the LM80C will find schematics and sources, useful even if they want to modify the project to create a version more suited to their needs. I decided to share all the information because it is thanks to the work shared by others that I was able to build the LM80C and I therefore decided to make available to others what I have learned and achieved in order to help others replicate what I have done.

So let's start from the beginning... If I remember well, it was Christmas 1984. Under our Christmas tree there was a COMPUTER! The Commodore 64, you say. No, it was a Commodore 16. Don't turn up your nose, please. The C16 was in part my own choice: I was interested in the 121 colours palette, which at the time was a very rare thing for a home computer, the C64 was too expensive for my family's means. So the C16 was my first computer. And I don’t regret that choice, I had a lot of fun with it and it also made me passionate for computers and computer science, a passion that has not left me for all the years to come... But back to C16, of course all of my friends had a C64 or MSX. Well, when I used to go their house to play games, during those carefree afternoons more than 30 years ago, I was always jealous of them, they had sprites! Those computers had sprites! Sure, I had 121 colours (and they were really cool on screen!) but I didn't have sprites. The games of that time had to fight the limit of 16 KB of RAM. These two factors indeed greatly limited the C16, a computer that was a resounding failure overall.

Years ago, when I discovered the Arduino platform, the love for 8-bit aroused in me. I started programming with the same “feeling” that I felt with the C16, namely optimizing the use of the resources and getting the most out of that little microcontroller. But I wasn't entirely satisfied. I didn't feel like I had a real computer on my hands. So in 2018 I decided to take it seriously, and I started studying the logic of the 74xx series and the most common 8-bit processors of the 80s still available. The choice narrowed around 3 candidates: the MOS 6502, the Motorola 6809 and the Zilog Z80. It was useless to go around it, I chose the latter, because of several factors: firstly, I wanted an 8-bit processor of that period but still easy to find on the market today, with extensive documentation and well supported on the network; secondly, I needed a good compiler, possibly a cross-platform one, so that I could compile the software (written in Assembly) with ease on my PC. After choosing the CPU, I laid the foundations for my system: I wanted 32 KB of

RAM and 32 KB of ROM, so I should have had no memory problems for software or the operating system; then I wanted graphics on my home TV, definitely in colour and with full sprite support; the sound had to match the rest of the hardware (I didn't want that 2-voice chip of the C16); and finally, it had to be able to interface easily with the outside world. The choice of the Z80 opened the way for me because I could draw inspiration from the MSX standard for a couple of solutions to be adopted for the rest of the computer: the graphics chip and sound chip of these systems are still available on eBay for a few euros, and they are easily interfaced to the Z80, and in fact I opted for the TMS9918A and the AY-3-8910. I also had a solution ready for the software: in the early 1980s the source of a complete BASIC interpreter for the Z80 was published and commented. Over the years it was then rewritten for modern compilers. I just had to (well, not exactly an easy thing to do...) adapt it to my hardware and, when everything was working, I started expanding it with new commands and features. Now my computer entirely operates like an 80s home microcomputer, with an old C16 keyboard (for nostalgic reasons...) managed directly by the machine, a graphics chip that generates a video signal that I send directly to my TV, and the sound is also generated by the computer and reproduced by the TV. Last but not least, there's a serial input/output port fully working to communicate with the outside world. Getting to all this was not easy, but it was not impossible either.

So let's start going a little deeper into the technical analysis of the system and the integrated circuits that I have used: this study will allow us not only to better understand how the 8 bit we own works but also to start thinking in a "bigger" way, that is, to lay the foundations for the development of our own system that, depending on the desire and knowledge, can reach the final realization of a complete computer.

Before starting to embed components on a breadboard or draw diagrams with KiCad, it is necessary to study thoroughly the structure of a common computer to understand how it is composed and how it works.

Figure 2 shows the block diagram of a computer:

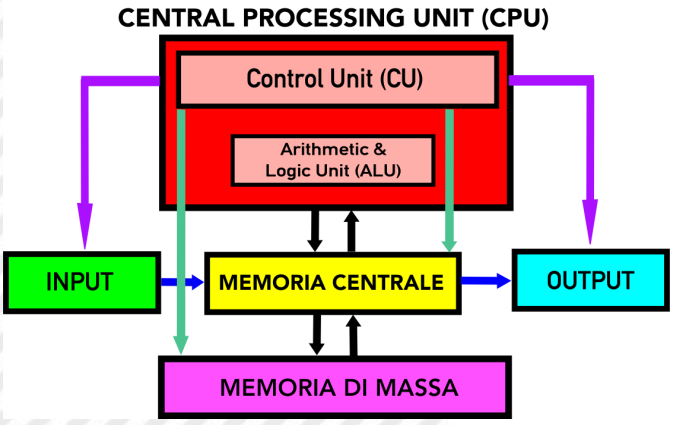


Figure 2: the block diagram of a computer.

There are 4 main and 1 secondary blocks. In order, the main blocks are: CPU, INPUT, OUTPUT and CENTRAL MEMORY. The mass memory is a secondary block because it is not vital for system operation and may not even be there.

Let's see in detail how each block works:

- **INPUT**: this is the block responsible for the management of incoming data. Input units such as a button or switch belong to this block. The keyboard (an array of buttons) is the most common input unit.
- **OUTPUT**: the block used to manage data outgoing from the system. Common output units belonging to this block are printers and the screen.
- **CENTRAL MEMORY**: this block contains the computer memory. It can be read-only (ROM), read-write (RAM), or both (most common).
- **CPU**: it's the "heart" of the computer. The Central Processing Unit (CPU) is the central data processing unit. It is composed by THE CONTROL UNIT (CU) and THE ARITHMETIC AND LOGIC UNIT (ALU). The CPU is the unit responsible for exchanging data: it manages the input data, processes the information communicating with the central memory to retrieve program instructions and/or to save intermediate processing data, and sends the data to the output devices so that the user can retrieve the processing results.
- **MASS MEMORY**: if there is a storage device outside the computer (such as a floppy disk or hard disk), the CPU also manages the reading and writing of data to this device for external storage and retention of data.

As you can see, a computer basically consists of a few functional blocks. The problem is not identifying the individual blocks: the "CPU" is our Z80 processor, for example, as the output is, for example, the video chip. The problem is to make all these different components work together. In fact, you need several integrated connection circuits (this is what the British call "glue logic") and in order to operate all these integrated circuits you need a minimum of knowledge of electronics to solve some problems that may arise. But none of this is insurmountable and with a good manual that explains the rudiments of electronics (what a resistor is and how it operates, what a capacitor is for, etc...) and with a little bit of integrated datasheet of the 74xx series we will succeed in our purpose.

Let's start with the technical features of the LM80C. Those who are more passionate about hardware than software will see that the computer is very similar to MSX 1, and in fact many design choices have been borrowed from those systems. Note: all components are easily available on the market, both as used parts (on online auction sites) and as new components, their purchase cost is modest and at a cost of a few tens of euros you can buy everything you need.

CPU: Zilog Z80 at 3.68 MHz
RAM: 32 KB
ROM: 32 KB

VRAM: 16 KB dedicated exclusively to the video chip
BASIC: integrated into the ROM, derived from NASCOM Basic, which in turn derived from Microsoft BASIC 4.7
Video: Texas Instruments TMS9918A, capable of generating a 256x192 pixel image with 15 colours (plus transparency) with up to 32 sprites (8x8 to 32x32 pixels)
Audio: Yamaha YM2149F (identical to General Instruments AY-3-8910), capable of 3 independent voices, 8 octaves, envelope management, white noise, and equipped with 2 8-bit input/output ports
Keyboard: 66 keys (for the record, I used an old C16 keyboard...), with independent cursor keys and function keys
I/O: Z80 SIO capable of handling 2 serial ports with speeds up to 57,600 bps; Z80 PIO equipped with 2 parallel 8-bit ports
Miscellaneous: Z80 CTC, timer/counter used to generate the serial clock as well as a system tick system with which I increase an internal clock and perform various operations at predefined intervals

Let's start from the heart of the system which, as seen above, is the CPU. As mentioned, we chose the Zilog Z80, which was released in 1976. The model we used is a CMOS version with a maximum clock of 6 MHz, well beyond the operating frequency of our system. In figure 3 you can see the pinout of the processor, with pins divided by function.

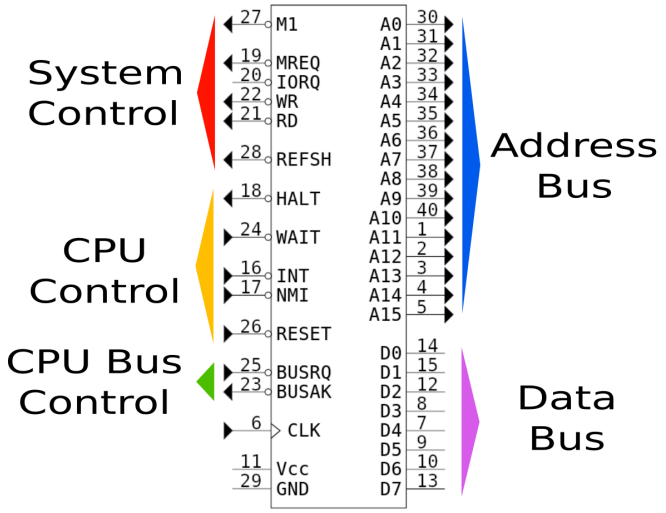


Figure 3: the pinout of the Zilog Z80 processor

The image shows that the address bus is 16 bits, so the maximum directly addressable memory is 2^16, or 64 KB. Similarly, the data bus works with 8-bit, as the internal architecture. In order to operate, the Z80 requires a single phase clock signal (a single line with a square wave signal) and a single +5 Volt power supply, as well as a memory from which to read the code to be executed once it has received the power. Then a circuit to give the reset signal to the processor must be added, which serves to reset all its internal registers just like when the computer boots. You can then see several other pins that we will not talk about for now and that we will deal with later.

Let's talk about the power supply circuit, shown in figure

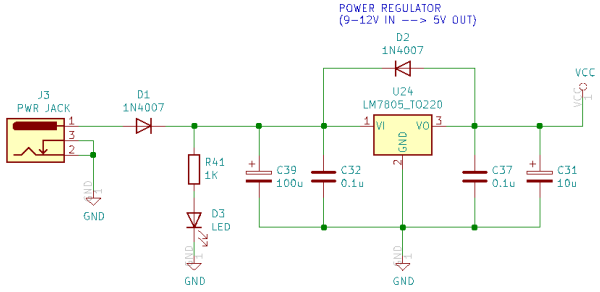


Figure 4: Circuit based on the LM7805

4: it is a circuit based on the classic linear voltage regulator LM7805, capable of providing a voltage of 5 Volts with a maximum current output of 1 A and which can regulate a variable input voltage in a fairly wide range: all we need to know is that any 9 or 12 Volt DC power supply is more than fine. I recommend that you equip the regulator with a cooling fin, especially if you plan to use a 12V input voltage: the greater the difference between the input and output voltages and the greater the heat it produces. At the LM7805 you can also replace an LM317: it is another very robust regulator whose output voltage is not preset as in the case of the 7805 but must be selected using a pair of external resistors. In the figure there are 2 diodes: D1, immediately after the jack, and D2, which connects the output pin and the input pin of the regulator. The diode D1 serves as the protection of the entire circuit, preventing that if the jack poles are accidentally inverted, the electric current does not flow in the reverse direction, damaging all the components. The diode D2, on the other hand, protects the output of the LM7805 from the voltage returns of the downstream capacitors when the power is disconnected: the capacitors are used as an energy "storage" to balance small fluctuations in the power supply. When the power is disconnected they discharge into the circuit: the diode D2 serves to prevent the current from entering the pin from which it usually exits and arriving inside the regulator by travelling in “backhand” ways, damaging the regulator itself, but passing through it towards the regulator input. In the circuit there is an LED on the input to see when the circuit is live.
Once the power supply is fixed, let's think about the

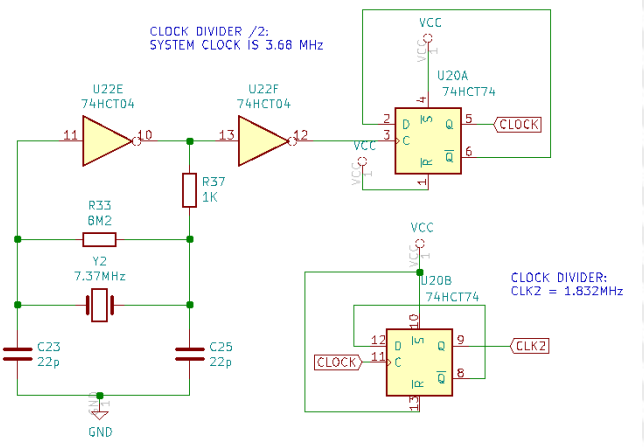


Figure 5: the system clock generation circuit

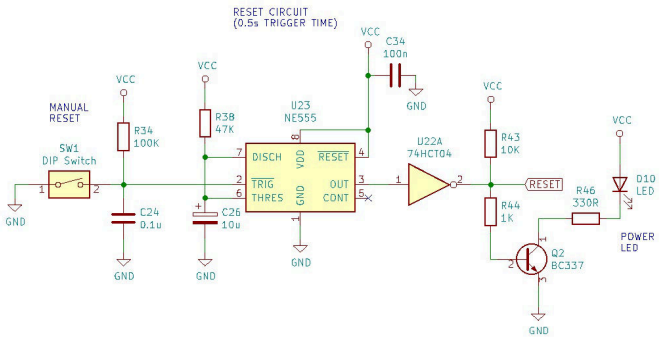


Figura 6: the reset circuit

system clock generation circuit. For the clock I opted for a 7.37 MHz quartz whose signal was divided by 2 to obtain an operating frequency of 3.68 MHz. This value is wanted because it is twice the frequency normally used to generate the serial clock, which is 1.83 MHz. On MSX 1 systems and many systems using the TMS9918A graphics chip the system clock was 3.58 MHz: this value was not chosen at random but was the result of a precise design choice to contain costs. The TMS9918A works with its own 10.74 MHz clock, which is triple the 3.58 MHz colour burst signal used to generate the colour carrier video signal. The TMS9918A also features this clock on one of its feet so to save another quartz many designers used this signal as a system clock. Since we do not need to contain costs, we have opted for a different solution that helps us elsewhere on the computer. The circuit used to generate the system clock and the serial clock can be seen in figure 5. Here I used 2 additional integrated circuits, a 74HCT04, which is a Hex inverter, i.e. a circuit that inverts the input signal (if it is “low” it makes it “high” and vice versa), and a 74HCT74, which is a double flip-flop. The inverter serves to stabilize the levels of the signal generated by quartz: being able to undergo slight fluctuations, the wave generated could have peaks that are not clearly recognizable by the CPU or by some other integrated because they are too far from the minimum required threshold: with the inverter I get a stable and always the same level. The flip-flip consists of 2 identical and separate circuits. Each part has been configured to operate as a divider for 2 of the input signal: that is, the diagram you see halves the frequency of the input signal. At the top the quartz frequency was increased from 7.37 to 3.68 MHz, which is the system clock, used by the CPU and the peripheral chips of the Z80 family that I used (PIO, SIO and CTC) while the bottom part further halves this value to 1.83 MHz, a frequency as mentioned used to generate the serial signal. Not only that, this value is also used clock of the audio chip, since it accepts a maximum clock of 2 MHz.

Figure 6 shows the reset circuit instead. Contrary to what you might think, even a good reset circuit is a fundamental component of a computer. In addition to sending the well-known signal that restores both the main processor and all the integrated peripherals to their initial state, the reset circuit also comes into play when power is supplied to the entire system: for the integrated ones to operate correctly they need a stable supply voltage close to 5 Volts. During the first ignition the voltage is not immediately available at this level but rises from zero: although quickly, there is a certain amount of time during which the voltage is below the minimum threshold. This value could cause the processor or one of the peripheral chips to operate abnormally, forcing a new reset to fix the problem. The reset circuit shown in figure 6 fulfils this purpose: thanks to a NE555, a very old but well-known integrated timer, the reset line is held at a high level for about half a second, a time more than sufficient for the voltage to stabilize in the entire circuit. After this time the reset signal is deactivated and the system can start normally.

If you look at the circuit, you will see another inverter present after the NE555: this is necessary because the timer activates with a high signal and then deactivates bringing the signal to a low level. But the reset of the built-in is recognized when the signal on these pins is at a low level: this way the NE555 would keep the chips always under reset. Using the inverter we invert the signal levels so that after starting the signal is low and, after the preset time has elapsed, it is brought to a high level, thus allowing the system to be initialized. Below the inverter there is a transistor used to turn on an LED when the reset signal is off: this way we can see when the computer is active.

Well, for this first article, I'd say that's enough. There are plenty of concepts and schemes to study and dive into. See you next time with the study of how the CPU interfaces with memory and peripheral chips, going to see the signals and components that are used to connect to memory (and select ROM or RAM) and to I/O devices.

Useful links

The LM80C project reference web page:
https://www.leonardomiliani.com/en/lm80c/
Electronic diagrams and firmware source code:
https://github.com/leomil72/LM80C
The Hackaday page dedicated to the LM80C:
https://hackaday.io/project/165246-lm80c-color-computer

Japan News
Game & Watch Vs MADrigal

by Michele Ugolini

Dear readers, please welcome Luca Antignano, also known as MADrigal. (see Figure 1)

We all knew this was the time.
We are experiencing a particularly dire year: 2020 is both a leap year and a lustre year.
In Italy we have now almost completely emerged from the planetary drama caused by COVID-19.

Moreover, we are experiencing an overcrowded period of mini consoles, remakes and more or less successful projects and, given the recent arrival of the "CoreGrafx mini PC Engine", I begin to think I am lost in the middle of Dante's 'path'.

In recent episodes I have talked about the conception of Game & Watch, the main personalities operating in the design room and, above all, the magical Japanese ingredients that have turned madness into objects that have now became sacred to us collectors. I have always defined Game & Watch as an electronic prodigy with a stylized immortal soul. It feels incredible to think of how many Japanese designers we should thank for their ideas that have become eternal. It is equally incredible to think about an equally talented genius in Italy, on the opposite side of the world, shares creative and brilliant ideals!

My infinite love for Japan is also motivated by this asynchronous spatiality, animated by a personality worthy of comparison, similar to the universe: generated by a slight and asynchronous fluctuation between opposite stages of matter.

Well, let's start the interview, today the honours go to the brilliant Luca Antignano.

RMW: "Luca, welcome, a warm virtual hug, tell us something about yourself and your nickname, your passion for G&W that I, like so many readers, share with you!"
LA: "Of course, thank you, I'm happy to share some background of my projects with your readers. I'm not exactly a youngster. I was born in 1974 and my first video game experience probably dates back to 1981, when I first saw a Pacman arcade cabinet in my town, Sassari. Since then, video games have become first an obsession, then a passion and finally a profession (albeit temporary). Now that I have stopped playing, due to the many commitments and the vicissitudes of my life, I still cultivate interest in video games, mainly retro. It's just one way to stay tied to a past lived with so many emotions. I've lived in Sydney for five years, where I work as an engineer. My nickname is a pretty random choice. Madrigal was the name of the elf character I used when I

Figure 1

played RPGs during my university years. It was simply a name that sounded good to me, had no particular meaning. When I started my programming project, I chose the nickname MADrigal, with capital 'MAD' almost reinforcing the 'madness' that I felt permeate my project. Read MADrigal as 'mad madrigal'.

The passion for G&W has deep roots for me. As a child I had never had any electronic games, however I desired them so much. I played with my friends' electronic games until 1984 when I received the C64 as a Christmas present from my parents. Since then, I have no longer been interested in those simple, trivial pocket games... Then, in 1999, I spotted a few handheld games for sale for cheap in a thrift store. I bought them and then wondered why there were no emulators available for these games. And I said to myself, well, I can try something, that would be fun. And so out of challenge and passion for the world of emulation (that in those years lived its most prolific phase) I decided to undertake my first project: '_MADrigal_'s Handhelds Simulators' (now changed more simply to 'MADrigal's Simulators'). (see figure 2)

Over the years I have actively collaborated to many projects such as MAME, Emuita.it, N! Zone (Nintendo Zone, which I founded and managed for several years), Zzap!Raine, Game & Watch Mania (my Italian website on G&W), Retroedicola Videoludica for which I created the Zzap! 2015 Special magazine and other publications."

RMW: "Let's get to the heart of the action, please tell us about the most important problems you have faced in your many projects."

LA: "I always start my projects by playing the games that I get from time to time. If the game is not fun, I put it away or trade it with others, hoping to find one (cheap) that is fun to play, and stimulating for me to programme.

Then I try the game for a long, long time, to try to fully understand its gameplay and see what happens in the various events (getting bonus lives or points, end of game, intermissions, etc.) and this is quite long, sometimes tedious, and complicated especially if the game is difficult. Once this has been done, I move on to acquire the game graphics, usually via a scanner.

Here we add another complexity. Sometimes games don't reveal all the graphics at the same time, so you have to scan it many times, and every time you hope to 'catch' a few more sprites. This is due to the fact that games with liquid crystal screens have the various segments on or off depending on the

Figure 2

occasion. At best, when the game is turned on, all the segments are switched on (test mode or 'ACL'), in other cases this does not happen and you have to turn the segments on a few at a time by playing with them. Or in the case of LED screens, you have to open the game, pull out the screen and scan it separately, which is laborious. (see figure 3)

After that, the scans need to be cleaned up and you need to edit the images to make them look real. These are all slow, laborious, and sometimes complicated procedures that go on for days. All this before you even start programming, so you can imagine how much effort is behind it.

A particularly difficult thing to replicate is the artificial intelligence of certain games. Apart from those where the game plays more or less randomly, there are others where for example ghosts chase Pacman, or your computer opponent defends or attacks you based on how you move. I programmed 3 games where I had to replicate AI, and it's always pretty complicated because you can't really 'invent' the AI, you have to programme it like the original game, so you have so many stakes and rules to follow. But it's good fun anyway!"

RMW: "Simulation, emulation, cloning: they are not synonyms! Would you like to describe these aspects better?"

LA: "You're right, they're actually so different, we could almost call them opposites! An interesting definition is given by the genius Nicola Salmoria in his thesis dedicated to MAME, in which he cites my project as an example of 'simulation' as opposite to 'emulation'.

Simulation, in this area, means not worrying about the hardware of games, but only about gameplay. It means playing the original game over and over until you know all its features, and then programming (for example with Visual Basic or Java) all the routines by recreating a game that resembles the original as much as possible. And that obviously also means using graphics as close as possible to the original. In this case I do not use the'ROMs' of the original game, but I create the routines of the game, programming them and inserting them into the simulator itself.

The difference is this: with simulators you download a single file, ready for use. With emulators, you need the ROMs to play.

Emulation means creating a program that reproduces the hardware (CPUs and other processors and devices) of the original video game. Once this 'virtual environment' is emulated, you run the program (the so-called 'ROM') that was typically created for that particular computer or game console. This is the case with MAME: you need ROMs because MAME is a collection of hardware emulators – but no software included. Without ROMs, it's like having a lot of computers, but no software to run them.

Broadly speaking, you can call a clone a simulator if you like. But you don't 'clone' anything, you program something that resembles it, whereas clone would mean 'identical', while the simulator will never be identical to the original. And neither will the emulator ever be 100%, no matter how close it gets."

RMW: "Can you explain to us the model by which you manage the interactions between the various

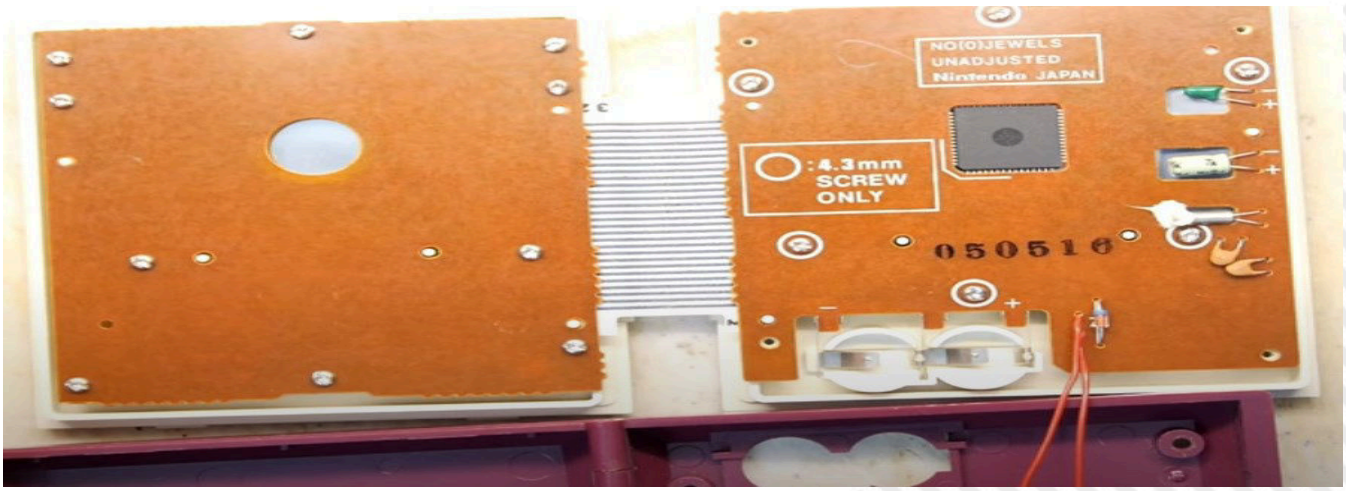


Figure 3

components involved? By what method do you manage to deal abstractly with each project and with what mechanisms do you manage the various interactions and/or user actions? Is the system replicable and, if so, how and with what possible limits from the point of view of both platforms and the user interface?"

LA: "Programming an electronic game simulator is no different from programming any other video game, at least at the concept level. Then of course there are specific features for the type of game. An electronic game has less components than traditional games: graphics is made of pre-shaped liquid crystal segments, for example a row of running men, each in a very precise position on the screen. You light one segment at a time, in its relative position, and it looks like a running man.

In the program, you simply worry about 'deciding' which segment to turn on and which to turn off, repeat this several times and combine it with keystrokes/joysticks so that it becomes interactive. Add the 'beep beeps' and the game is ready. To do this, you prepare in advance the routines and 'matrixes' in which you locate the various segments, this makes it possible to connect the sprites (the segments) to a mathematical model usually simple enough to manage with a coordinate system.

The characteristic of electronic games is that there are no 'collisions' between elements. When you make a mistake playing, it's usually because you haven't moved your character to a safe position within a set time. Time is marked by 'beeps', so you more or less know how much time you have to make your move and if you don't, a miss is marked. It seems complicated to explain, but very easy once you see it running or in a video.

The system is certainly replicable, the chosen programming language or environment (mobile phone, computer or console) do not matter: there are simulators that you can play by pressing the buttons of the game on a touch screen (to replicate the feeling of plastic games with buttons) or on the keyboard or joystick. This depends on how the simulator was programmed."

RMW: "Code preparation, image preparation, audio preparation, randomization management, optimization, every project remains an impressive amount of work on the table. Without a rigid, indeed, I would venture to say 'Japanese' organizational style, it would seem impossible to merge the many Japanese figures who gave birth to a single G&W. Yet in the MADrigal project you managed, almost alone, to give birth to dozens of wonderful works (see Figures 4 and 5). Please tell us about your style!"

LA: "You're right, today we're used to large programming teams, graphic designers, musicians, directors and producers. But for an electronic game, you don't need all this. Having not only programmed classic electronic game simulators but also created new electronic games (for an American video game company), I was able to experience, as an insider, the production technique that led Nintendo to produce the first pocket LCD games. The process needs software engineers, graphics, 3D modellers and one or more game designers.

In the case of simulators, I had to learn things by myself, combining programming (in my case Borland Delphi) with graphic editing (with Paint Shop Pro usually) and audio (Audiowave or similar). I follow

Figure 4

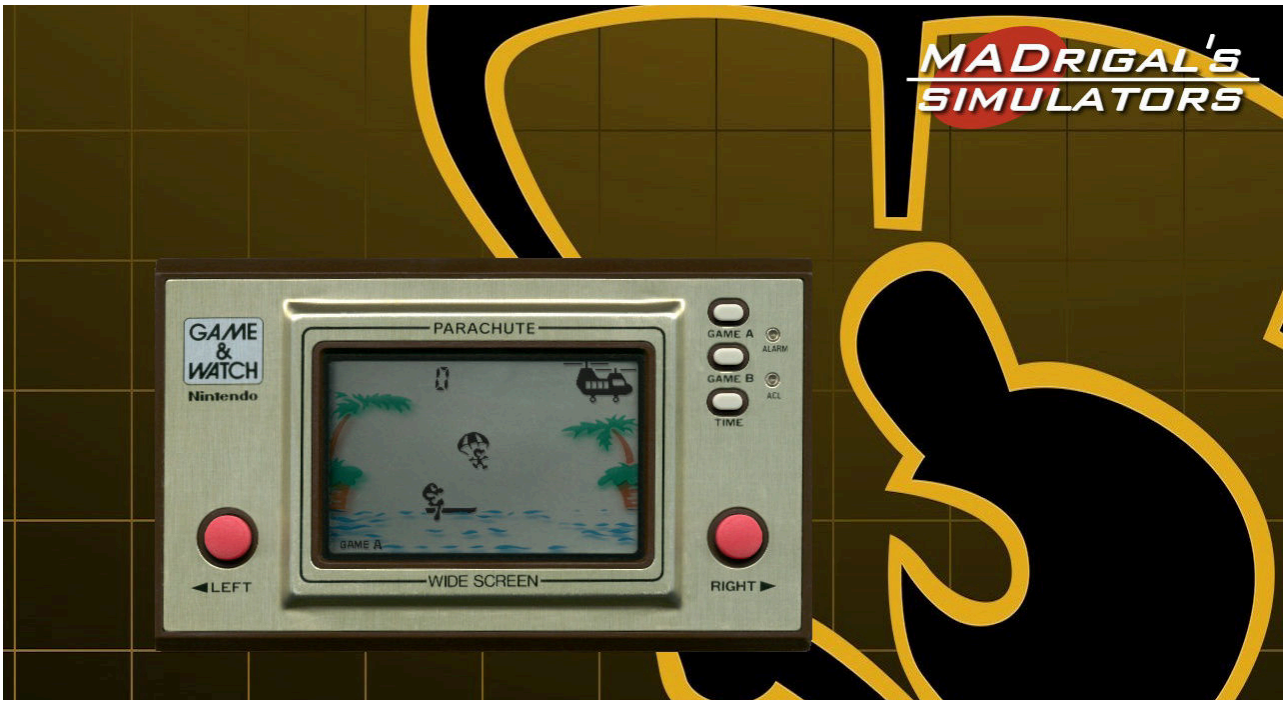


Figure 5

more or less the same process most times: I try the game, scan the graphics, prepare the images and place them on the design window, and then programme the input routines – that is how the buttons work. Then I play over and over, record the audio and insert the various sound effects into the programming environment. Lastly, I program the game routines and test the final product until I realise that I have inserted all the possible combinations and procedures present in the original game.

My style is that of absolute realism. I create the graphics so that they are identical to the original game, with shadow effects on the LCD screen, animated buttons, and so on. Sound effects are always samples recorded from the original game. I really like the idea of having not just the playable LCD screen, but also the plastic casing around, with animated buttons."

RMW: "We were born and raised with the (at the time) all-powerful C64, power was in our hands. We are in 2020, a period of video games animated by impressive graphic levels, do you think that today there are products with dubious souls?"

LA: "Well, I think the video game market has evolved just like every other market in the past centuries. Everything begins with a niche market, with such a pioneering spirit, and then gradually becomes a large market in which it is difficult to invent completely new products, but a new product 'enhances' an existing one. The first period of video game history is full of ideas, but also failures. The best ideas have survived and today are the basis of

Figure 6

the new video games. We find a little bit of Pacman, Donkey Kong, Tetris and Space Invaders in every video game, albeit in different forms.
It is difficult today to criticise the video game industry, because in the end the really good products sell, those with poor quality eventually succumb and we will not remember them anymore. I welcome the various 'C64 mini' or 'NES mini', in the end they are a bit the result of nostalgic operation, a way to keep our past alive and make it known to the new generations. Of course, these are commercial transactions, but let's not forget that the video games industry moves billions every year, not unlike the twenty Marvel or Disney movies. Yes, in the end we can get tired, but we can also ignore the new products and take refuge in the previous ones, such as movies or video games from a few years ago, always fun."

RMW: "How many different G&W and electronic handheld game projects have you completed, how many are you working on, and how many would you still like to work on?"
LA: "I've programmed 60 electronic game simulators, actually 59 plus one that is a special version of Donkey Kong, based on the original game logic, but sporting 4 different sets of graphics and sound, each customizable by the user. An authentic homage and act of love for what I think is the most emblematic pocket game ever.
I am not working on any projects, I have stopped for a few years, following my move to Australia, the new job, the new life and the many challenges that I face every day – not least the COVID situation, the impossibility of returning to Italy to see my family, the work that changes shape and other personal things. I mean, as you can see, this is not the time for me to programme video games. I do not deny that I would like to be in a more serene condition that gives me incentives to plan new game designs. In that case, my first choice would be 'Mario's Cement Factory' from Nintendo – an electronic game that I love and that would be a challenge to programme."

RMW: "I imagine that you own many G&W and handheld games of many colours, brands, clones, etc. Are you looking for any particular game to buy? Maybe some readers can help you with your hunt!"
LA: "Until a few years ago I had a lot of games, but I never called myself a collector. I bought or traded games mainly for the purpose of programming simulators, rarely bought them just for fun. I gave away or sold almost everything, once the games were programmed, or decided that I would never programme them (for various reasons), I had no problem giving them away.
I'm not looking for any games right now. Sometimes I'm offered free games under the condition to programme its simulator – but I always refuse. My time right now is worth more than getting a game for free."

RMW: "There are so many nice G&W-themed memes on the web. Can you tell us a funny anecdote that amazed you about your past work?"
LA: "The most peculiar anecdote is when I was contacted by one of the largest American game producers, and I was asked to work for them to design and produce electronic game demos – which were then actually produced and sold in USA. I have programmed 4 games including 2 under Namco

Figure 7

license. Of these, 2 were produced. It happened around 2007 (if I'm not mistaken). It was interesting and stimulating."

RMW: "CreatiVision, IntelliVision, ColecoVision, just some of the brilliant consoles of the 80s. How does your commitment arise, especially with CreatiVision?" (see Figure 7).

LA: "I was a greedy reader of the Italian classic videogame magazine, 'Video Giochi' before becoming the lucky owner of a C64. So I knew the many classic consoles of those years well, except the CreatiVision, which I had never been able to try but it seemed really cool.

When I realised in 1999 that there were emulators available for basically every classic console, except CreatiVision, I talked to a couple of friends about it. One had an original CreatiVision, and the other was a very skilled programmer of emulators. We joined forces and soon the CreatiVEmu project was born. First as an attempt to emulate the console, and then as a conservation project that collects scans, ROMs, games, books. Then in 2007 I produced, with a friend, the first multi-cartridge for CreatiVision. And in 2009, we produced a diagnostic cartridge. Since then, I have kept the project active, adding the information I find on the internet from time to time. The project is enormous and is the world reference point for that console."

RMW: "So much passion, so much love for these fantastic projects, in addition to the donations that you can make through your site:
http://www.madrigaldesign.it/sim/
have you ever thought about working with large companies such as Sony or Microsoft or something to market your genius?"

LA: "Well, I never really wanted to think about it seriously. I like the idea that mine is a free project, a hobby. I have worked in the IT, engineering, education, graphic and advertising sectors for many years. I have concluded that programming must just be a hobby for me. I have made my professional choice in favour of building engineering. Carrying out projects in the field of video games requires time, concentration, study and continuous evolution, I think that at my age and after the various changes in my life, it is better to stay anchored to my current career.

But I don't deny that I'm glad to see that several programmers have taken an interest in my work and dedicated heart and soul to projects that made it possible to 'transplant' my games on modern platforms such as Andriod, iPhone, Sony, Microsoft, Raspberry, Switch, NES Mini and so on. Now all my games are playable on virtually every platform on the market. That makes me really happy."

RMW: "What could be the future evolution of this project? Do you have any future plans?"

LA: "At the moment I have no other plans, except to keep alive what I have already done so far and support new programmers who contact me for support or to expand what I have already done. Same with the CreatiVEmu project. In the future you never know..."

Well dear readers, we thank you for your attention and above all we thank MADrigal for answering the questions of the interview.

If you want to deepen the discussion you will find a lot of amazing stuff at:
http://www.madrigaldesign.it/

In addition, the site contains three large projects. N! Zone: archive of the website, closed in 2006 but now available, all in Italian (see figure 6). Game & Watch Mania: dedicated to G&W, also entirely in Italian. CreatiVEmu: active project, in English, with huge databases, emulators, ROMs, forums and everything CreatiVision-related.

Lastly, I recommend that you keep your interest on G&W alive, in fact if things go back to normal and global production starts again, there will soon be great surprises that I would like to talk about in the next issue or the following one.
See you soon!

Can we multiply the number of games in THEC64? Yes, we can!

by Marco Pistorio

The project "THEC64", appreciated by many and mistreated by many others, continues to go ahead with inflated sails.

Moreover, it is news from a few days ago that a version of the device that traces the characteristics and the real size of the VIC-20 Commodore and that will be called "TheVIC20" is about to be launched. This proves that Retrogames Ltd is still focusing on the project and is therefore producing a further variant of "THEC64", already on the market in the "Maxi" and "Mini" versions.

This article will not deal with the goodness of the project itself, the fact that "THEC64" must necessarily be an object pleasing to collectors of the real "commie" or not, nor will I insist that, in my opinion, anything that can revive the emotion of playing with a Commodore 64, that still brings the Commodore 64 to the forefront by presenting it to old and above all to new users is something that deserves, regardless of everything, to be always and in any case accepted and promoted.

In this article, I will focus on the titles available in "THEC64" and particularly on how to make them become almost ten times more than the 64 normally available.

In the past, I have read something about it, but all the solutions proposed went to permanently modify the device and forced physical access to its internal hardware, obviously voiding any kind of guarantee.

The solution I am talking about today focuses on making 623 titles available and does not involve any invasive intervention or permanent modification of "THEC64" and works with both models of "THEC64" on the market: the "THEC64" Maxi and the Mini. Not bad, is it?

The idea is roughly this:
A USB flash drive with games, game snapshots etc. is provided along with an update of the "fake" firmware of "THEC64". The device is switched on, the false update is applied and, when it is turned on again, we will find the new games in a new carousel.

The new carousels are actually 4.
A maximum of 226 games can be hosted in each carousel.
There is a first carousel of favorite games.
Then a carousel of games whose titles start with a character "0 to G", with 222 games, a carousel of games whose titles start with a character "H to R", with 219 games and finally a carousel of games whose titles start with a character "S to Z" with an additional 182 games, for a total of 623 games.

You can navigate from one carousel to the next by moving to the "wrench" icon and then pointing them at the name of the carousel on which to move and choosing to load it.
After any session of gaming, turn off the device and safely remove the USB flash drive.
Restarting the device will return everything as the fabric's setting, without having made any permanent changes!

You can download the folder containing all the already packed games following the link on the project page "Project Carousel USB by Spannernick", at this address: https://thec64community.online/thread/501/project-carousel-usb
Only a simple registration in the forum "THEC64 Community" is needed.

Extract the content of the .zip file into a USB flash drive and... The game is done :)
PLease note that the project is FREE only for non-commercial use.

Personally, I find everything very comfortable and intuitive.

There is also the possibility to change the content of the games made available thanks to a tool located inside the folders that contain each carousel.

The executable name to launch is "TheC64MaxiGameTool.exe"

For more details on how to customize the various carousels, add new ones and much more I suggest you read carefully the contents of the official page that you can reach by following the link already provided.

Have fun and greetings to all!

Back to the past... episode nr. 2: Windows 2000

by Marco Fiaschi

Welcome to this exciting virtual journey through the operating systems that have kept us company for half a decade of our lives. Surely you're wondering why this second stage ends in Windows 2000, rather than Windows ME (the infamous Millennium Edition).

This tour of ours, in fact, is not bound to the versions of the Windows kernel, but to the release date of the different OS software. Before introducing Windows 2000, I want to answer a few questions I received following the publication of the article about Windows 98 virtualization in RetroMagazine World #22-IT (for those who have missed it, I invite you to go to the official website to download a copy, although in Italian only, fot now).

Here are some of the most common questions:
Q - Why does Windows 98 have bad graphics on Virtualbox?
A – Unfortunately, the drivers offered by Virtualbox (GuestAdditions) are not available for the Windows 9x operating system suite, so you need to install third-party drivers directly on the emulated Windows.

Q - Why is Windows 98 emulation very slow?
A – The answer is in the question. An emulation is still an emulation. It is unreasonable to get the same performance that we can get from a physical disk installation of an operating system. In the virtualized system settings, we try to enable at least 3D graphical acceleration, bringing the dedicated memory to the maximum allowable (128MB or 256MB).

Q - Which video driver is best selected in the settings?
A – Virtualbox provides as many as 4 drivers that will allow the operating system to recognize dummy graphics cards. They are: VboxSVGA, which is the optimal choice for Windows 7 and later operating systems. VMSVGA is chosen by default when dealing with Linux operating systems, and is an emulation of the Vmware SVGA driver (built right into the Linux kernel). Finally, we find the VboxVGA driver, which is the last beach to choose from, because it is a very outdated driver supported by Windows Vista and earlier. This means that for the operating systems examined in these articles (Windows 98, Windows 2000 and Windows ME that you will find in the next issue of RetroMagazine World), it is the most suitable driver that you can choose. Be careful, though. It's just that by choosing this driver, the graphics of the systems will increase to 32bit. You still need to install external drivers.

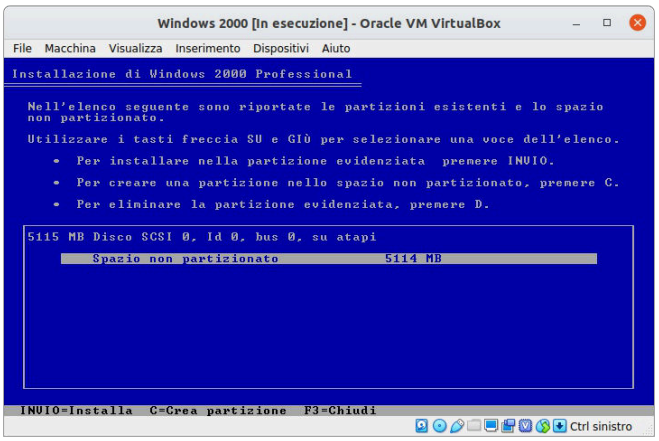


Figure 2 - OS installation in progress

At the end of this little detour with Q&A, we leave for the second stage of our journey. Released on February 17, 2000 under the names Odyssey and then Neptune (or rather NT 5.0), Windows 2000 was an operating system belonging to the Windows NT (New Technology) family, based entirely on 32-bit hybrid kernels, whose support ended on July 13, 2010. Windows 2000, unlike the 95, 98 and ME editions, was not designed as a "home" operating system, or intended for clients, but was designed to meet the needs of professionals and businesses. Windows 2000 does not come from the Windows 9x family, but from the NT family, to which all Windows editions for servers and workstations belong, including NT 3.1, NT 3.5, NT 4.0, until the current Windows 10.
This operating system introduced for the first time ever a new type of filesystem, NTFS version 3.0 (next releases will later constitute a standard still widely used), Encrypting File System (EFS) support, a new type of kernel completely rewritten based on the Windows NT 4.0 predecessor in terms of graphical interface and network protocol management, Active Directory, and Background Intelligent Transfer System (BITS). Windows 2000 has been widely used in the company, but very poorly distributed among consumers, due to its total incompatibility with video games. Simply put, this system was ideal for running professional graphics, mathematics, CAD and workstation programs (high-performance computers with very low

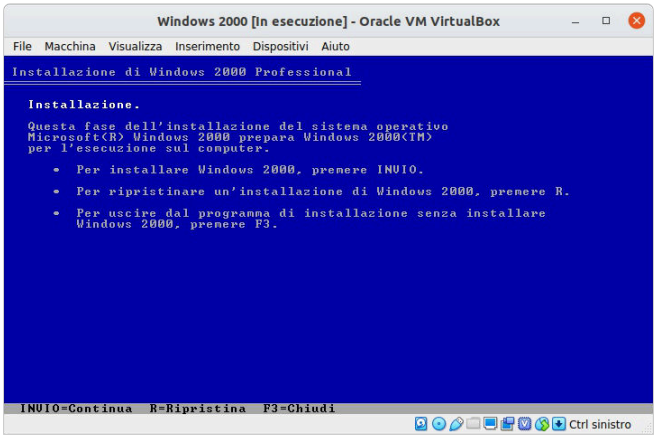


Figure 1 - Let's start with the installation

Figure 3 - Formatting the hard disk partition

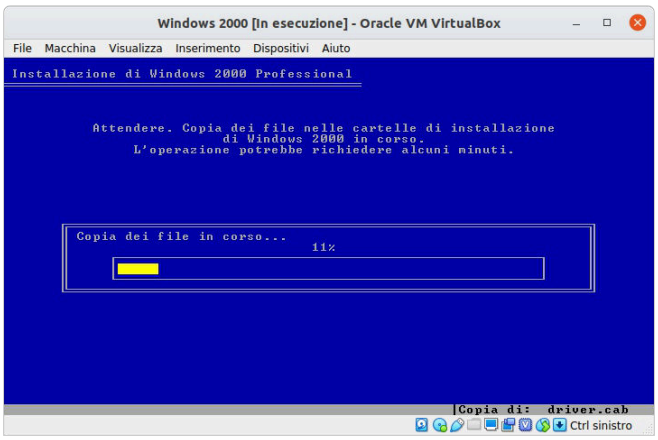


Figure 4 - File copy of Win2k in progress

CPU latency to maximize processor power to the last Hz). Windows 2000 was released in 4 versions, including Windows 2000 Server, Advanced Server, and Datacenter Server. Almost unknown are the 64-bit versions of Windows 2000 developed for Intel's Itanium and Itanium 2 processors. Windows 2000 is still considered one of the safest operating systems ever developed by Microsoft... It's no coincidence that Windows 10 is considered the same today, and the Pro for Workstation edition is similar to Windows 2000 the most because of some features implemented in the operating system. A further curiosity of this operating system is its version of the kernel which I remember being the fifth, the other consumer systems such as Windows 95, 98 and ME are developed with kernel version 4. You can upgrade Windows ME to 2000, but not the other way around, even though 2000 came out before ME.

After this extensive introduction to the system, let's quickly see how to install Windows 2000 on our Virtualbox.

Click on the item "New" in the software menu, then enter the name to give to our virtual machine, choosing "Microsoft Windows" under the item "Type" and "Windows 2000" under the item "Version". VirtualBox now recommends 168MB of RAM to be allocated to the virtual machine, but we will allocate 512MB for more efficient emulation. We then choose the type of file to use for the virtual hard disk that will be dynamically allocated for our convenience VDI (Virtualbox Disk Image).
When the software asks us for disk space, we will enter the value of 5GB (although Windows 2000 requires less than 700MB), but this will allow us to benefit from better emulation. Having done this, we have created the virtual machine that we can start with a double click on its name.

Figure 5 - Windows 2000 Professional booting

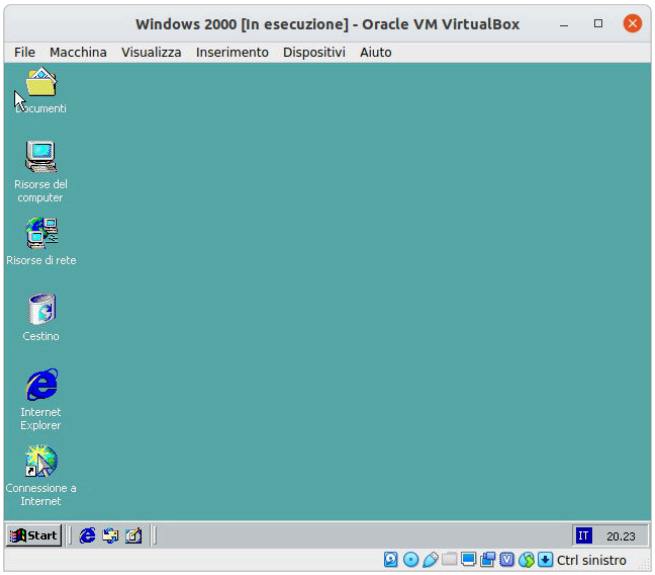


Figure 6 - Windows 2000 running in the VM

We select the host reader from which to boot the operating system and proceed with the installation (figure 1) which will begin with loading all the drivers and files necessary for the system. We click Enter when Windows prompts us, accept the Microsoft license agreement, and then click Enter again to select the partition that the installer suggests (Figure 2).

Little curiosity... Have you noticed that the installer name is "Installing Windows 2000 Professional"? Don't worry, you haven't got the wrong edition and/or version! There's no Home edition of Windows 2000, but there is only one and that one is Pro.

In the next screen (figure 3) we format the partition with the new filesystem introduced by Microsoft, NTFS and wait for the copy process of the Windows 2000 files on our virtual hard disk to finish (figure 4). Once the system is restarted, we will immediately notice a very new graphics of the installation of the operating system that is still used for installation PACKAGES.MSI.
Next we will have to set the date and time, the keyboard layout, and then we will have to enter the user name, the product key (available in the Windows 2000 manual), the computer name, the password and finally wait for the installation to complete.
When we least expect it, a window will appear informing us that the installation of Windows 2000 has been successful and completed. When the system is restarted, you can enjoy the new Windows 2000 boot screen that reads "Developed with NT technology" (Figure 5), and then the system will be ready for use (Figure 6).
We will immediately see a refresh in the desktop icons, although we will not be able to benefit from eye-catching graphics because Virtualbox, as mentioned above, does not support these operating systems (unfortunately support begins with XP), unless you install third-party drivers.

Unfortunately the journey to the second destination ends here, but I invite you to look for some websites from which you can download endless software and abandonware games...

It's true that Windows 2000 wasn't designed to work with video games, but trying it never hurts, does it?
See you the next episode ;-)

Amstrad CPC - Redefining the character set

by Francesco Fiorentini

Background
A few days ago, I came across a post by Arturo Dente on Retro Programming Italia (RPI) by The Nerds, containing a magazine in pdf format for the benefit of the whole group. Arturo had wanted to share that particular magazine in order to urge other members of the group to retrieve a piece of basic program for Commodore 64, useful to load into memory an LM code to split the screen into two parts, text and hi-res... A very useful feature if you want, for example, to write a text adventure with some accompanying graphics.
Driven by curiosity, I open the post and see that someone else, part of the RMW editorial staff, David La Monaca, had already taken care of this operation.
I still open the magazine to inquire about the code, but I am immediately attracted to another program that is a couple of pages after the one indicated by Arturo.

Future Set on Amstrad
The name strikes me immediately, leaving little room for imagination. This is almost certainly a set of instructions for redefining the Amstrad CPC character set to change its look & feel. A quick look at the code, immediately avoids any doubt. It is a real redefined character set to use in our programs in Locomotive Basic.

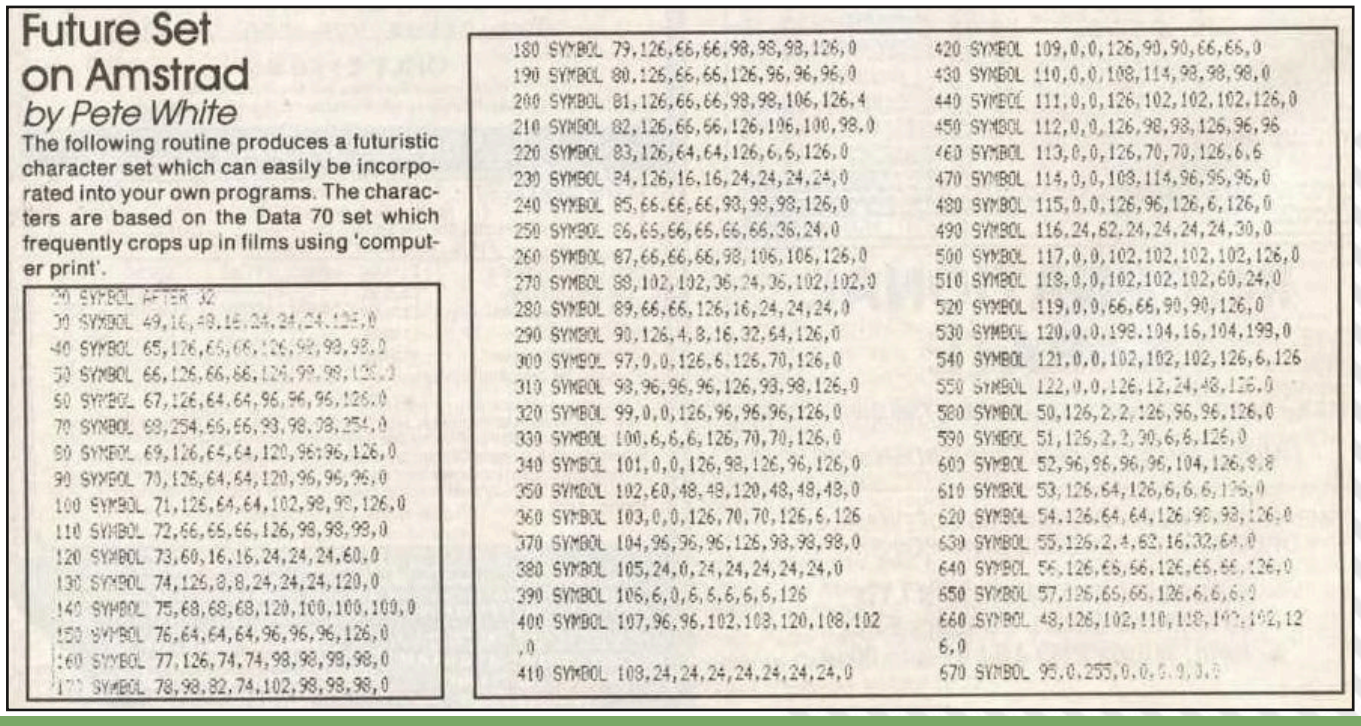
I had an idea! In number 23 we ported the text adventure "Astronave Farmer" to Locomotive Basic. This is a game that sees us aboard a starship adrift in space... What better chance to take advantage of a futuristic character set? I decide to work on the recovery of this piece of history, initially authored by Pete White, to give it a second life.

Code retrieval
Obviously if the code could have been copied and pasted from the attached pdf, Arturo would never have asked for the support of other users of the group... So, looks like, the only solution was to rewrite the code from scratch. But I'm extremely lazy and the idea of writing one by one about eighty lines of code with tons of numbers has never even touched me. In addition, many of the numbers are almost unreadable. It's 2020, is it possible there's no more practical way?

Of course, there is! An OCR, a character recognizer! I therefore decide to try this route.
I look for a free character recognizer online and I choose for: https://ocr.space/.
With the snipping tool I create 3 images, one for each column of the code and load immediately the first one into the OCR.
The first result is disappointing beyond expectation. Very few recognized numbers and lots of unreadable text.
I some adjustments to the settings and opts to use the OCR 2 engine which, according to the site, is more efficient in number recognition.

This time the result is acceptable. Approximately 60% of the text was acknowledged. I just need to make substitutions in the text to correct the commas that had been interpreted as periods, manually correct the wrong numbers, and finally correct all the wrong occurrences of the 'SYMBOL' command. Hard work, but faster than writing everything from scratch.

Have we done here?
I corrected all the code, I'm satisfied!



Future Set on Amstrad
by Pete White

The following routine produces a futuristic character set which can easily be incorporated into your own programs. The characters are based on the Data 70 set which frequently crops up in films using 'computer print'.

```
30 SYMBOL AFTER 32
40 SYMBOL 65,126,66,66,126,98,98,98,0
50 SYMBOL 66,126,66,66,126,98,98,126,0
60 SYMBOL 67,126,64,64,96,96,96,126,0
70 SYMBOL 68,254,66,66,98,98,98,254,0
80 SYMBOL 69,126,64,64,120,96,96,126,0
90 SYMBOL 70,126,64,64,120,96,96,96,0
100 SYMBOL 71,126,64,64,102,98,98,126,0
110 SYMBOL 72,66,66,66,126,98,98,98,0
120 SYMBOL 73,60,16,16,24,24,24,60,0
130 SYMBOL 74,126,8,8,24,24,24,120,0
140 SYMBOL 75,68,68,68,120,100,100,100,0
150 SYMBOL 76,64,64,64,96,96,96,126,0
160 SYMBOL 77,126,74,74,98,98,98,98,0
170 SYMBOL 78,98,82,74,102,98,98,98,0
180 SYMBOL 79,126,66,66,98,98,98,126,0
190 SYMBOL 80,126,66,66,126,96,96,96,0
200 SYMBOL 81,126,66,66,98,98,106,126,4
210 SYMBOL 82,126,66,66,126,106,100,98,0
220 SYMBOL 83,126,64,64,126,6,6,126,0
230 SYMBOL 84,126,16,16,24,24,24,24,0
240 SYMBOL 85,66,66,66,98,98,98,126,0
250 SYMBOL 86,66,66,66,66,66,36,24,0
260 SYMBOL 87,66,66,66,98,106,106,126,0
270 SYMBOL 88,102,102,36,24,36,102,102,0
280 SYMBOL 89,66,66,126,16,24,24,24,0
290 SYMBOL 90,126,4,8,16,32,64,126,0
300 SYMBOL 97,0,0,126,6,126,70,126,0
310 SYMBOL 98,96,96,96,126,98,98,126,0
320 SYMBOL 99,0,0,126,96,96,96,126,0
330 SYMBOL 100,6,6,6,126,70,70,126,0
340 SYMBOL 101,0,0,126,98,126,96,126,0
350 SYMBOL 102,60,48,48,120,48,48,48,0
360 SYMBOL 103,0,0,126,70,70,126,6,126
370 SYMBOL 104,96,96,96,126,98,98,98,0
380 SYMBOL 105,24,0,24,24,24,24,24,0
390 SYMBOL 106,6,0,6,6,6,6,6,126
400 SYMBOL 107,96,96,102,108,120,108,102
,0
410 SYMBOL 108,24,24,24,24,24,24,24,0
420 SYMBOL 109,0,0,126,90,90,66,66,0
430 SYMBOL 110,0,0,108,114,98,98,98,0
440 SYMBOL 111,0,0,126,102,102,102,126,0
450 SYMBOL 112,0,0,126,98,98,126,96,96
460 SYMBOL 113,0,0,126,70,70,126,6,6
470 SYMBOL 114,0,0,108,114,96,96,96,0
480 SYMBOL 115,0,0,126,96,126,6,126,0
490 SYMBOL 116,24,62,24,24,24,24,30,0
500 SYMBOL 117,0,0,102,102,102,102,126,0
510 SYMBOL 118,0,0,102,102,102,60,24,0
520 SYMBOL 119,0,0,66,66,90,90,126,0
530 SYMBOL 120,0,0,198,104,16,104,198,0
540 SYMBOL 121,0,0,102,102,102,126,6,126
550 SYMBOL 122,0,0,126,12,24,48,126,0
560 SYMBOL 50,126,2,2,126,96,96,126,0
590 SYMBOL 51,126,2,2,30,6,6,126,0
600 SYMBOL 52,96,96,96,96,104,126,8,8
610 SYMBOL 53,126,64,126,6,6,6,126,0
620 SYMBOL 54,126,64,64,126,98,98,126,0
630 SYMBOL 55,126,2,4,62,16,32,64,0
640 SYMBOL 56,126,66,66,126,66,66,126,0
650 SYMBOL 57,126,66,66,126,6,6,6,0
660 SYMBOL 48,126,102,110,118,102,102,12
6,0
670 SYMBOL 95,0,255,0,0,0,0,0
```

I run the program and notice that some characters are unreadable... Yet it seemed to me that I had correctly interpreted the numbers, even the almost illegible ones. Obviously, I was wrong, or there were mistakes in the initial code as well.
I must therefore correct the wrong characters. I point my browser to the address http://xlr8.at/8x8hexbin/ and check the characters that don't appeal to me.
After a few tests, I'm satisfied by my job and I declare my recovery work completed.

Astronave Farmer Enhanced

I renumber the lines of the code I retrieved and add them at the end of the 'Astronave Farmer' code. Using a GOSUB (go to subroutine) I run the part of the code that redefines the characters before the program prints something on the screen and... Here it is! The text adventure has its 'Future Set' character.

The game Astronave Farmer Enhanced can be downloaded from the RetroMagazine World website at this address:
www.retromagazine.net/download/Farmer_Enhanced.dsk

The game is listable, so you can safely check its code, but if you want it in txt format, contact me.

Oh, I forgot. It's not over. The enhanced version is also equipped with a splash screen in SCR format. Want to know more?
Don't miss the next article.

Here's the redefined character code.

```
10 REM ******************************************
11 REM * Future Set on Amstrad CPC
12 REM * original code by Pete White
13 REM * Popular Computing Weekly 7-13 August 1983
14 REM *
15 REM * Typed and corrected by
16 REM * Francesco Fiorentini on June 2020
17 REM * RetroMagazine World July 2020
18 REM ******************************************
20 SYMBOL AFTER 32
30 REM Upper case chars
40 SYMBOL 65,126,66,66,126,98,98,98,0
50 SYMBOL 66,126,66,66,126,98,98,126,0
60 SYMBOL 67,126,64,64,96,96,96,126,0
70 SYMBOL 68,254,66,66,98,98,98,254,0
80 SYMBOL 69,126,64,64, 120,96,96,126,0
90 SYMBOL 70,126,64,64,120,96,96,96,0
100 SYMBOL 71,126,64,64,102,98,98,126,0
110 SYMBOL 72,66,66,66,126,98,98,98,0
120 SYMBOL 73,60,16,16,24,24,24,60,0
130 SYMBOL 74,126,8,8,24,24,24,120,0
140 SYMBOL 75,68,68,68, 120,100,100,100,0
150 SYMBOL 76,64,64,64,96,96,96, 126,0
160 SYMBOL 77,126,74,74,98,98,98,98,0
170 SYMBOL 78,98,82,74,102,98,98,98,0
180 SYMBOL 79,126,66,66,98,98,98,126,0
190 SYMBOL 80,126,66,66,126,96,96,96,0
200 SYMBOL 81,126,66,66,98,98,106,126,4
210 SYMBOL 82,126,66,66,126,106,100,98,0
220 SYMBOL 83,126,64,64,126,6,6,126,0
230 SYMBOL 84,126,16,16,24,24,24,24,0
240 SYMBOL 85,66,66,66,98,98,98,126,0
250 SYMBOL 86,66,66,66,66,66,36,24,0
260 SYMBOL 87,66,66,66,98,106,106,126,0
270 SYMBOL 88,102,102,36,24,36,102,102,0
280 SYMBOL 89,66,66,126,16,24,24,24,0
290 SYMBOL 90,126,4,8,16,32,64,126,0
295 REM Lower case chars
300 SYMBOL 97,0,0,126,6,126,70,126,0
310 SYMBOL 98,96,96,96,126,98,98,126,0
320 SYMBOL 99,0,0,126,96,96,96,126,0
330 SYMBOL 100,6,6,6,126,70,70,126,0
340 SYMBOL 101,0,0,126,98,126,96,126,0
350 SYMBOL 102,60,48,48,120,48,48,48,0
360 SYMBOL 103,0,0,126,70,70,126,6,126
370 SYMBOL 104,96,96,96,126,98,98,98,0
380 SYMBOL 105,24,0,24,24,24,24,24,0
390 SYMBOL 106,6,0,6,6,6,6,6,126
400 SYMBOL 107,96,96,102,108,120,108, 102,0
410 SYMBOL 108,24,24,24,24,24,24,24,0
420 SYMBOL 109,0,0,126,90,90,66,66,0
430 SYMBOL 110,0,0,108,114,98,98,98,0
440 SYMBOL 111,0,0,126,102,102,102,126,0
450 SYMBOL 112,0,0,126,98,98,126,96,96
460 SYMBOL 113,8,0,126,70,70,126,6,6
470 SYMBOL 114,0,0,108,114,96,96,96,0
480 SYMBOL 115,0,0,126,96,126,6,126,0
490 SYMBOL 116,24,62,24,24,24,24,30,0
500 SYMBOL 117,0,0,102,102,102,102,126,0
510 SYMBOL 118,0,0,102,102,102,60,24,0
520 SYMBOL 119,0,0,66,66,90,90,126,0
530 SYMBOL 120,0,0,198,104,16,104,198,0
540 SYMBOL 121,0,0,102,102,102,126,6,126
550 SYMBOL 122,0,0,126,12,24,48,126,0
555 REM Numbers
560 SYMBOL 48,126,102,110,118,102,102,126,0
570 SYMBOL 49,24,56,24,24,24,24,126,0
580 SYMBOL 50,126,2,2,126,96,96,126,0
590 SYMBOL 51,126,2,2,30,6,6,126,0
600 SYMBOL 52,96,96,96,96,104,126,8,8
610 SYMBOL 53,126,64,126,6,6,6,126,0
620 SYMBOL 54,126,64,64,126,98,98,126,0
630 SYMBOL 55,126,2,4,62,16,32,64,0
640 SYMBOL 56,126,66,66,126,66,66,126,0
650 SYMBOL 57,126,66,66,126,6,6,6,0
680 SYMBOL 95,0,255,0,0,0,0,0,0
1000 CLS
1011 PRINT "Future Set on Amstrad CPC"
1012 PRINT "original code by Pete White"
1013 PRINT "Popular Computing Weekly 7-13 Aug 1983"
1014 PRINT ""
1015 PRINT "Typed and corrected by"
1016 PRINT "Francesco Fiorentini on June 2020"
1017 PRINT "RetroMagazine World July 2020"
```

Figure 1 - The redefined characters of 'Future Set'

How to make a splash screen for Amstrad CPC in SCR format

by Francesco Fiorentini

This article is a logical continuation of the previous one. I preferred to make two articles rather than one, for the convenience of readers, who might be interested, even over time, in a single topic.

A brilliant idea

After editing the character set of the Astronave Farmer game, I realized something was still missing.

The idea of a text adventure set in space is interesting, the redefined characters undoubtedly add a pinch of pepper to the game, but in my modest opinion there was still something missing to customize it effectively.

It would be nice to add, above the description of the places visited, a small image that depicts the locations and thus transform a completely text adventure game into a text adventure with graphics.

Obviously, such a project is rather ambitious and takes time to be successfully completed. Not to mention that I am almost unable to draw...

In addition, that my knowledge of Amstrad CPC's graphic is zero and you will have a fairly faithful idea of the ambitious goal that had crossed my mind.

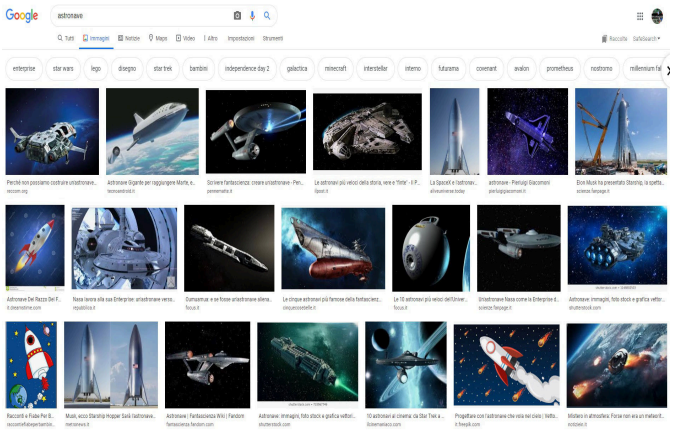
I therefore decide to proceed in stages. This article depicts the stahge 1: adding a static image as a splash screen while loading the game.

Starship

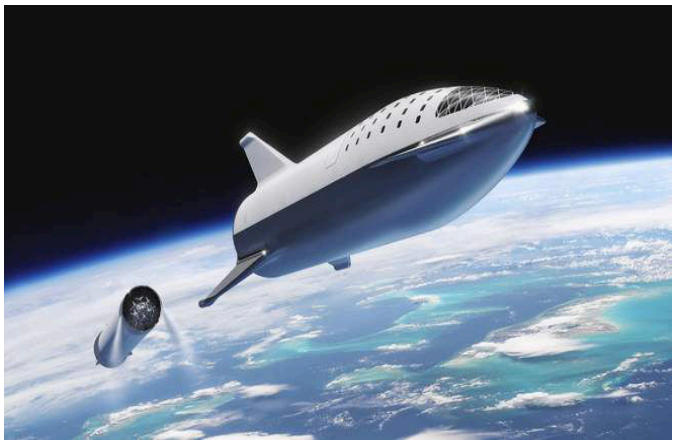
Okay, now that I had defined an approach, the least was done. All I had to do was draw an image that somehow represent a spaceship in the deep space.

Easy to say, but rather difficult to do in practice if, like me, you are truly denied in drawing and the maximum you can aspire to is the stylized representation of any real object.

But it's 2020 and we can easily find billions of images with a simple Google search. I open the browser and simply type the word 'spaceship'. Immediately dozens of beautiful images appear in front of me



They are all beautiful, but the second image got all my attention. I decide it will be the splash screen of the game 'Astronave Farmer Enhanced'.



First problems

Great, we have a candidate for our intro image, but... The image is A JPG with a resolution of 660 x 371 pixels, a resolution of 96 dpi and a color depth of 24 bits. Not a problem nowadays, but far too much for our Amstrad CPC.

I need to find a way to turn a JPG image into something that our Arnold (as he was affectionately known in the 1980s) can digest and possibly display on video.
I do some research and it looks like the format par excellence is the SCR. The SCR format is nothing more than the physical representation of the Amstrad CPC memory that can be reloaded using the Locomotive Basic LOAD command.

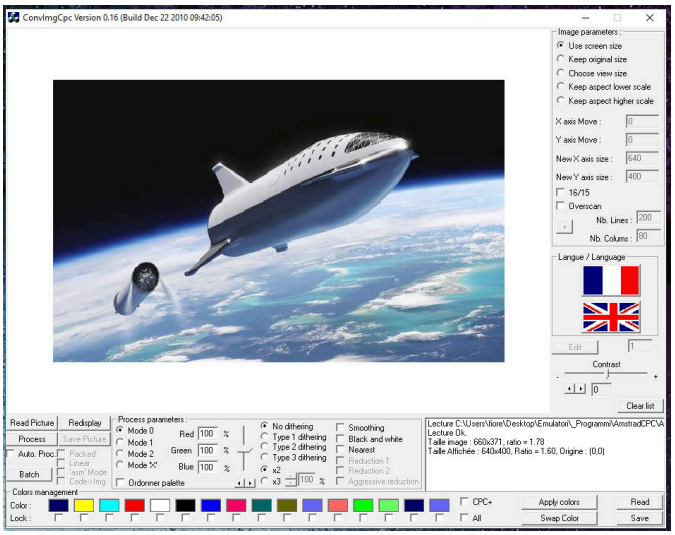
My search for a converter leads me, among others, to a program called ConvImgCPC.

ConvImgCPC is a freeware program created by Demoniak that allows the conversion of images from PC to CPC.

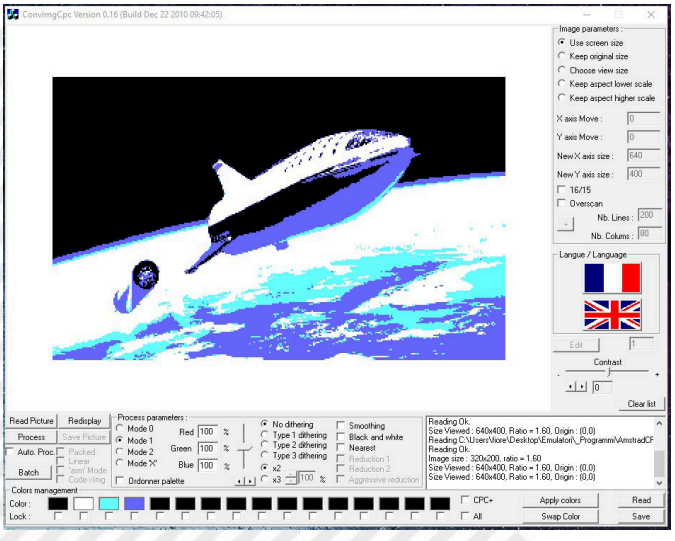
Unlike many other similar products that I found, mostly command-line, this is accompanied by an excellent graphical interface, rather intuitive and with the ability to choose an impressive number of options to generate the SCR images.

After choosing the English language and having uploaded the image in JPG format, I start to play with the many options.

I want to generate an image to be displayed in MODE 1 with a resolution of 320x200 and with only 4 available colors. I must admit that the software really does what it promises. With just a few clicks we can transform a complex JPG image into a SCR file ready to be displayed by the CPC.



Everything is really simple: select THE MODE 1 format in the Process Parameters menu and then click on the Process button to process the image. The software will do everything on its own, adapting the image to the correct size and taking care to convert colors in a fairly faithful way.



There are dozens of options to manage the colors, but for what I wanted to do, the automatic process is more than enough. We are simply generating an image with 4 colors.

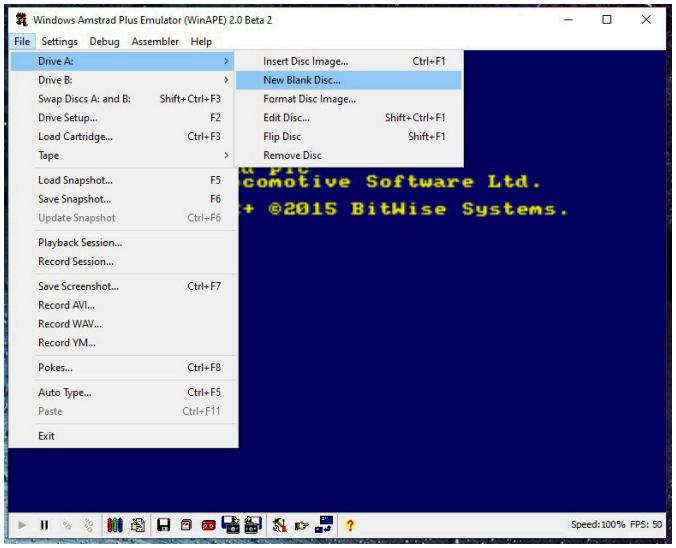
To save the image, simply press the Save Picture button and once you have chosen the SCR format, give the file a name. You can also export the color palette using the Save button in the Colors Management menu. I must admit, I still figure out what to do with the palette... You can probably use it in advanced graphics programs such as GIMP. Or with some Assembly code even on the Amstrad CPC.

SCR image on Amstrad CPC

Well, now we have a SCR image, but how to upload it to a .dsk virtual disk of the Amstrad CPC?

The following instructions will examine the WinAPE emulator, an excellent Amstrad CPC emulator.

Once WinAPE is launched, we proceed to create an image of a disk with the option -> File -> Drive A: -> New Blank Disc. Be sure to choose the directory where to create the image of the disk and assign it a valid name.



After creation, the disk must be formatted using the command -> File -> Drive A: -> Format Disc Image and answering OK to the next question.

Now that the disk is formatted, let's try accessing its contents using the CAT command.

If everything was done correctly, we should receive the following message from the emulator:

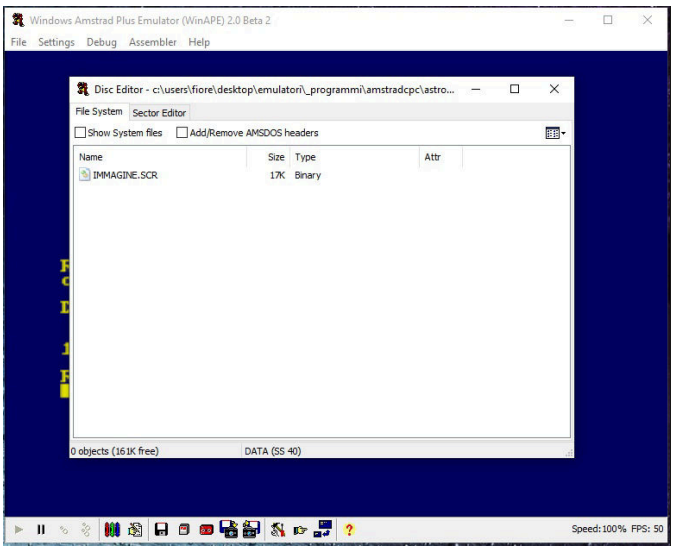
DRIVE A: user 0

178K Free

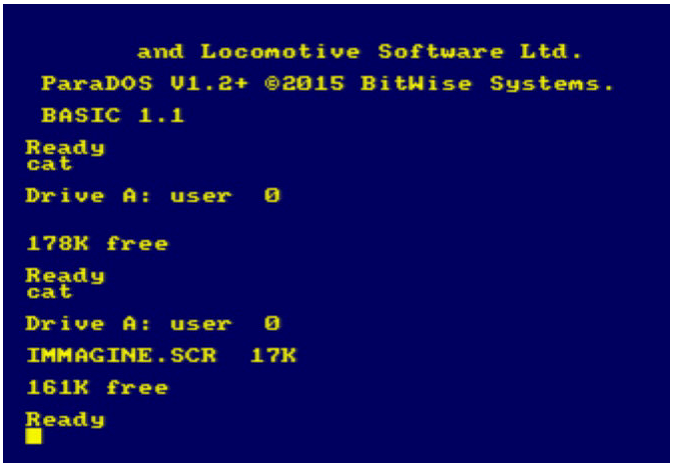
Do you see that? Perfect, let's copy the image to the new disk. Once again, WinAPE makes this task very easy because we can edit the content of the disk directly from Windows

using a convenient emulator command.

From the -> File -> Drive A: -> Edit Disc menu, simply drag the file you want to copy onto the disk, in this case the image, and that's it!



Run the CAT command again. This time we should find the image file within the disk.



Let's display the SCR image

Now that everything is ready, we can finally display the image on our Amstrad CPC.
Write this little program in Locomotive Basic:

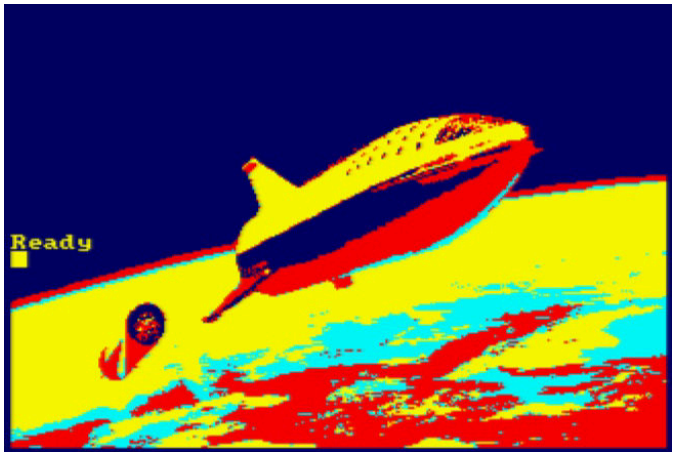
```
10 LOAD "SCR IMAGES", &C000
```

and give it RUN...

Hey, what's going on? What are these colors?

The image we had generated with ConvIMGCPC was slightly different. Why do we now see these 4 colors instead of those displayed by the converter?

To understand what happened we need to take a closer look at the SCR format...



The SCR file is nothing more than a map of the pixels in memory indicating which of the four MODE 1 colors has to be used.

To understand how these pixels are mapped we need to study the Amstrad CPC 'graphics card', the Motorola 6845 Cathode Ray Tube Controller, friendly known as the CRTC.

The CRTC

The standard graphical modes supported by the CRTC are:
MODE 0 - 160x200 - 16 colours
MODE 1 - 320x200 - 4 colours
MODE 2 - 640x200 - 2 colours

If we check the pixel representation in the 3 modes, we notice that:
- in MOODE 0 each byte contains 2 pixels (4-bit) 16 colors
- in MOODE 1 each byte contains 4 pixels (2-bit) 4 colors
- in MOODE 2 each byte contains 8 pixels (1-bit) 1 color

The peculiarity is that the pixel distribution is not linear, but follows the following structure:

	Bytes/Pixel structure						
VM (Mode)	7	6	5	4	3	2	1	0
% 00 (0)	A0	B0	A2	B2	A1	B1	A3	B3
% 01 (1)	A0	B0	C0	D0	A1	B1	C1	D1
%10 (2)	A	B	C	D	E	F	G	H

Starting with mode 2, the simplest one, each byte of memory contains 8 pixels of 1-bit color depth. The pixel representation in this case is simple, 1 bit for each pixel starting from the most significant.
In the case of mode 1, the one that interests us, each byte

contains 4 pixels, with a 4-bit color depth. As can be seen in the above table, the distribution in this case is not linear. The color of the first pixel consists of bit 7 and bit 3.

In mode 0, each byte contains only 2 pixels and the distribution is even more complex, as the color of the first pixel is composed of bit 7, followed by bit 3, bit 5 and bit 1...

Taking a look at the standard Amstrad CPC palette it will be immediately clear why the colors displayed by the emulator are different from those of the converter...

The first 4 colors are exactly what we see in the image posted shortly before.

0 – Dark blue (1)
1 – Yellow (24)
2 – Cyan (20)
3 – Red (6)
4 – White (26)
5 – Black (0)
6 – Blue (2)
7 – Magenta (8)
8 – Dark cyan (10)
9 – Dark yellow (12)
10 – Light blue (14)
11 – Pink (16)
12 – Green (18)
13 – Light green (22)
14 – Dark blue (1)
15 – Cyan-blue (11)

Let's change the colors

But we are stubborn and want at all costs to display the image with the 4 colors that best characterize it.

Try typing these 4 lines of code:

```
10 INK 0.0
20 INK 1.26
30 INK 3.14
40 LOAD "SCR IMAGES", &C000
```

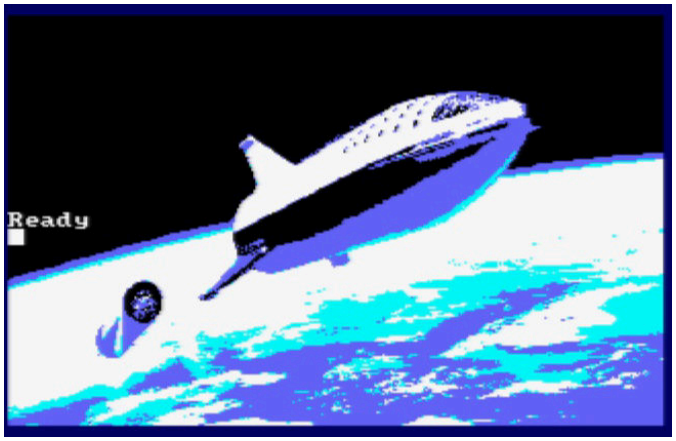
Save the program to disk with the command:

```
SAVE "LOADER.BAS"
```

and restart the emulator with a nice RESET.

Now launch this command:

```
RUN "LOADER.BAS"
```



Much better, do you agree?

But there's still something we're not entirely convinced of. That Ready message is extremely annoying. Ultimately our intention was to create a splash screen for the Farmer game, so let's get rid off of it.

Let's use our Loader again, this time really to launch the game FARMER.BAS.

Copy the game you find in the repository of Antonino Porcino to this address: https://github.com/nippur72/8-bit-projects/tree/master/astronave-farmer
add (if you want) the characters of the previous article, save everything as FARMER.BAS and then write this simple program:

```
10 INK 0.0
20 INK 1.26
30 INK 3.14
40 LOAD "SCR IMAGES", &C000
50 FOR I=1 TO 5000
60 NEXT I
70 RUN "FARMER.BAS"
```

Et voila! The Astronave Farmer Enhanced's game with the splash screen is ready!

I hope this simple guide can help some of you create games for Amstrad CPC with minimal graphics.

As always, if you have any questions, please do not hesitate to contact me.

Abbreviations & shortcuts on using a Graphical Interface

by Alberto Apostolo

During the study of Data Analysis and Business Intelligence using a well-known spreadsheet, I often had the need to abbreviate some sequences of data commands with a mouse and keyboard.

I looked up on line tricks to replace a phrase like "Left-click and drag the AutoFill handle from cell A2 to cell A10" with an abbreviation of type
CnD([Fill handle].[A2:A10]).

But not finding anything satisfactory, I decided to create "motu proprio" an abbreviation system that would also help in the use of other software with G.U.I. (Graphical User Interface).

Notations using the mouse and for keyboard pressed keys
The most frequently performed actions using the mouse are as follows:
1) Click (left button press)
2) Right-Click
3) Double-Click (double left click)
4) Drag&Drop (select, drag and drop)
5) Click&Drag
6) Mouse Hover (move the mouse over graphics).

N.B.: Some authors consider actions 4) and 5) synonymous. There's actually a distinction [MA20]. The Drag&Drop action requires two graphic elements: the first is the element to be dragged, the second is the "target" on which to release. The Click&Drag action requires a graphic element and an offset consisting of one or more parameters. Here some examples of Click&Drag : changing the width of a window, filling cells automatically, adjusting the sound of the computer speaker by moving a slider like the one shown in Figure 1.

The actions mentioned above can be abbreviated with acronyms (some of them found on the Web):
1) LMB (left mouse button)
2) RMB (right mouse button)
3) DCLK (double click)
4) D&D (drag and drop)
5) C&D (click and drag)
6) MH (mouse hover).

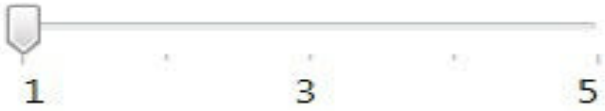


Figure 1

There are several notations to indicate the pressing of one or more keys. One of these is the new Microsoft notation (e.g. Ctrl+X) that is quoted on Wikipedia but everyone can freely adopt the most congenial notations.

The graphics components or widgets
In Computer Science, you define a widget (window gadget) as any graphical component of a graphical user interface (Wikipedia source).

Some widget types are listed below (the explanation can also be found on Wikipedia):
1) Containers: window, modal window, etc.
2) Navigation: toolbar, link, tab, etc.
3) Commands: menu, ribbon, dock, icon, button, etc.
4) Dialogue: dialog box, dialog box file,alert box,about box, etc.
5) Input: check box,text box,combo box,list box,drop down list,slider,spinner, etc.
6) Output: status bar,label,tooltip,progress bar,grid or data grid,etc.

The types listed above can be enriched by adding cells, matrices, vectors, graphs, sheets, etc.
"Everything is a Widget" is the philosophy adopted by some open-source software that creates native interfaces for iOS and Android systems.

Start building notation
A widget has as its identifier an alphanumeric string W. If you also want to indicate its type T then the notation T.W is adopted. Because the focus is exclusively on navigating through the various components of a graphical interface, other features such as colors , shape, size, screen position, etc., are overlooked.
N.B.: Even W.T notation is fine (similar to the representation "FileName.Ext"). Strings (unless otherwise stated) are not case-sensitive.

As in a sort of screen scraping, to express the A action on a widget group W1,..., Wn uses a function-based notation (A is an alphanumeric string as well)
A(W1, ..., Wn) .
If a W output widget is produced, it will be written
W=A(W1,..., Wn) .
If strings (used to identify widgets, types, actions) contain characters other than letters and numbers then they are enclosed in square brackets as in SQL. Another "loan" from SQL is the comments preceded by a double "dash".

To indicate a Wk widget contained in the W widget you write W\Wk, using the character "\" as in specifying the path of a file.
The asterisk * is used to highlight a widget on which an action is mandatory (e.g. a field to edit in a questionnaire). If the context requires it, it can also be used for actions. However, it is recommended that you add the asterisk only where it is necessary to avoid weighing the writing down.
Examples:
1) Click(icon.Save)
2) DnD(W1,[Recycle Bin]) -- Drag&Drop to "target"
3) Edit([E-mail Address]*) -- action on a required field
4) Edit(dialogbox.Price\textbox.Amount)
5) [Ctrl+Shift+Enter], [Ctrl+Click]
6) Press*(anykey) -- mandatory action

N.B.: The key combinations pressed are a special case of actions and can exist even without parameters.

Operators to chain actions

I have not studied the paradigm of functional programming. The choice of function-based notation is due to my mathematics studies and the ability to intimately use function composition to describe more structured actions. In addition, some "operators" are available to write "algebraic" expressions with round brackets: "|" (Alt+124) indicates actions to be performed in the sequence in which they were written, "/" (shift+7) exclusive alternative, "+" actions that can be performed in any sequence.
The use of the "|" character to concatenate sequential operations is similar to the UNIX operating system and is used with the same meaning in some texts for the European Computer Driving Licence (ECDL) certification.

Example: Creating a vector in a spreadsheet by specifying a cell-range.
Select(cell.[A2: A5]) | [Ctrl+Shit+Enter]

If A, B, C are actions on widgets, the following properties apply:
1) |, /, + are associative i.e.
(A x B) x C = A x (B x C),
2) / and + are switchable i.e.: A x B = B x A,

Create a dictionary of actions and types

Given the wide freedom in choosing action names e widget types, if you want to share notes with other people, it is wise to create a small “dictionary” with short explanations.
The proposed lists are designed not to use square brackets in delimiting strings (who wants can make additions to the “dictionary”).

Actions:
1)Click oppure LC (Left Click)
2) RC (Right Click)
3) CC (Click Click = Double Click)
4) DnD (Drag and Drop)
5) CnD (Click and Drag)
6) CnP (Copy and Paste)
7) XnP (Cut and Paste)
8) Select (selecting object)
9) Edit (edit cell,textbox, ecc.)
10) Copy
11) Cut
12) Paste
13) Close (a window)
14) Press (a key)
15) MH (mouse hover)

Actions (for spreadsheet):
16) Delete (eliminating cells with sliding of those nearby)
17) Clear (deleting only the contents of cells)

Types: window, modalwindow, toolbar, link, tab, menu, ribbon, dock, icon, button, dialogbox, filedialogbox, alertbox, aboutbox, checkbox, textbox, combobox, listbox, dropdownlist, slider, spinner, statusbar, label, tooltip, progressbar, grid , datagrid, matrix, vector, cell, fillhandle, plot, sheet, key.

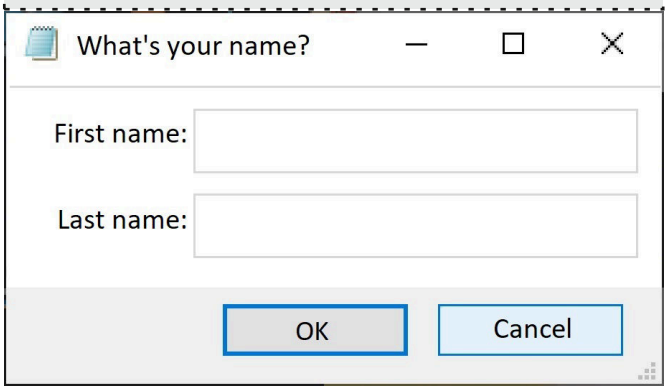


Figure 2

What is a cell-range

In a spreadsheet, a regular cell-range is a matrix or a vector consisting of the top left cell (minimum value) up to the bottom right cell (maximum value) [PX20].
Examples: C7 (a single cell), E2:E7 (vertical vector), A3:F3 (horizontal vector), A1:C6 (matrix of 6 rows and 3 columns).
An irregular cell-range is the union of separated regular cell-range groups with the character ";" (i.e. A1:C6;C9;E2:E7).

Screen Scraping

It is achieved when one application interacts with another by sending commands (pressed key, etc.) and “reading” the answers like a human being [HT18].

3) | is not commutative i.e. A | B other than B | A.
Operator | is distributive only to the left of / :
A | (B / C) = (A|B) /(A|C).
The + operator is distributive with respect to /:
A+ (B/C)=(A+B)/(A+C)=(B+A)/(C+A)=(B/C)+A.

Example (describe object editing in Figure 2):
Edit(What's your name?]) =
(Edit([First name]) + Edit(t[Last name]) |
(Click OK/ Click Cancel)

N.B.: Someone might object that actually you could "click" on "OK" before you have even edited the two required fields. The objection is good but the goal is to describe at least one sequence of sensible actions and not analyze all possible events in the use of an interface.

Groupings

In some special cases, conventions can be used to obtain more synthesis.
If the widgets W1, ..., Wn are subject to the same action A and x operator, you can write:
A(W1) x... x A(Wn) = A (W1 x... x Wn)
If the widgets W1, ..., Wn have the same type T and are subject to the same action A and x operator, you can write:
A(T.W1) x... x A(T.Wn) = A (T.(W1 x... x Wn))

Example (editing the object again in Figure 2):
Edit(What's your name?]) =
Edit([First name] + [Last name]) |
Click(OK / Cancel)

Document a widget structure

Sometimes it is useful to make a rudimentary documentation of the structure of a widget.

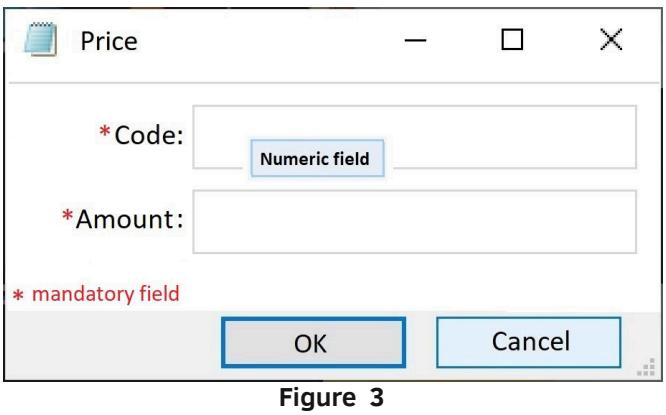


Figure 3

W notation (W1,...,Wn) means that W depends on the W1,...,Wn widgets while W notation {W1,...,Wn} means that W is composed of the W1,...,Wn widgets.

Example: There are two mandatory boxes in the Price dialog box. A tooltip (consisting of a label) warns Code is numeric (Fig. 3).
Price{textbox.Code*,
textbox.Amount*,
tooltip.Msg(Code){label.[Numeric field]},
button.OK,
button.Cancel
}

If it's not too complicated you can group widgets by type as follows:
Price{ textbox.(Code*, Amount*),
tooltip.Msg(Code){label.[Numeric field]},
button. (OK, Cancel)
}

N.B.: In [HT18] in Chapter 2, I found that this representation has a certain similarity to the "Data Language" and the "Resource Files".

Editable Widgets

In the case of editable widgets (such as spreadsheet cells) it is convenient to be able to assign (in square brackets) a default value or an expression.
Examples:
1) Edit([First Name]=[Mario] + [Last name]=[Rossi])
2) Edit(cell.A7=[ROUND(A1+A4;2)])

N.B.: In example 1, "+" is an operator and counts as a separator. In example 2, "+" belongs to an expression within the square brackets.

Conclusions

Existing symbols and User Experience have been reused without pretending to invent yet another computer language (in my opinion, there are enough languages). The goal is to immediately have notes in text format to describe in a fairly orderly manner the navigation through the various menus and windows of a graphical interface. For those who wish to contribute with improvements, it is worth inviting to present them on the pages of RetroMagazine.

Bibliography

[CZ05] M.Coggi, C.Zanon, "ECDL. Eserciziario e simulatore completi.",Edizioni FAG Srl, 2005.
[Gas08] C.Gasparini, "ECDL CAD. Il manuale", Apogeo Editore, 2008.
[HT18] A.Hunt, D.Thomas, "The Pragmatic Programmer" (Edizione Italiana printed by Apogeo, 2018).
[MA20] AA.VV., "Drag and drop steps", https://help.mabl.com/docs/drag-and-drop-steps , consulted on 27/4/2020.
[MH05] J.Moran, V.Hull, "Brilliant ECDL", Pearson Education, 2005.
[PX20] AA.VV, "What is a Cell Range", https://www.perfectxl.com/about-spreadsheets/Excel-glossary/what-is-cell-range/ consulted on 28/4/2020.

The 1st RMW 8-bit Computer Chess Tournament

by David La Monaca and Mathias Lorenz

Presentation

In a previous issue of RetroMagazine we reviewed two 8-bit chess programs (namely, Cyrus Chess I on ZX Spectrum 48K and Colossus Chess 3.0 on Atari 800XL) and we made them play against each other in a quick match of 6 games. We also took stock of the playing strength of the most popular chess engines in recent years. For the sake of completeness, in the same article we introduced AlphaZero, the AI software by DeepMind/Google, which in a quite surprising way appeared on the market and proved to be able to defeat without appeal all the traditional chess engines such as Stockfish, Komodo, Deep Shredder, Houdini, Fritz, etc. How did it accomplish this hard task? By beating Stockfish 8 (one of the strongest engines around, scoring about 3400 ELO points) in December 2017 in a match of 100 games. Alphazero won 28 times, 25 of which playing with White, and drew the remaining 72 games.

The second clash between these two monsters of calculus, which can be safely defined "generational", took place a few months later and the results were even more surprising than the first match. It also marked a real point of no return, definitely opening the doors to a new era in computer chess and in computer science applied to AI. For the record, an updated version of AlphaZero literally crushed Stockfish 8 in a match of 1000 games with an extraordinary score of +155 -6 =839 (155 wins, 6 losses and 839 draws). And that's not all, because AlphaZero also won against the traditional engine in other matches arranged with different thinking times (even with a difference of up to 10 to 1). Some more matches saw the new AlphaZero sonorously beat the latest development version of Stockfish, namely the version 9, with results very similar to those of Stockfish 8, according to DeepMind. The machine learning engine also won all the games against a variant of Stockfish that used a very extensive opening book. The addition of the opening book, however, brought some help to Stockfish, which eventually won a considerable number of matches when AlphaZero played with Black, but not enough to win the match.

As said, in order to indulge an old curiosity of mine, in issue 5 of RetroMagazine (April 2018) an article/review on a couple of 8-bit chess programs had turned into a 6-game match. The challenge between Cyrus Chess I (Spectrum 48K) and Colossus 3.0 (Atari 800XL) ended 3.5 to 2.5 in favor of Colossus, who showed that showed an overall much stronger algorithm, substantial control during the middle game and a lower tendency to make mistakes than his opponent. Some computer magazines of the 80s and 90s tried to establish which 8-bit chess program was the strongest. The Colossus series, culminating in the 4.0 version, seemed to be the most credited candidate for this award, but no one has ever given us any proof.

After some time, last May 2020, my piece was republished in English in issue #00 of RetroMagazine World and was noticed by Mathias Lorenz, a retrocomputer enthusiast and chess player. After a quick email exchange, we decided to try and define the most efficient 8-bit chess program ever. We wondered how and with what rules could we achieve such a result, assuming it is possible to achieve an absolute result. After some further considerations, we decided to arrange a full microcomputer chess tournament. The 8-bit software challengers were carefully selected among all those published in the 80s and at the end 4 of the best selling programs were admitted to the tournament. Each program was combined with a microcomputer, also selected among the most popular of the time. Here is the complete list of participants:

Amstrad CPC 464 / Colossus 4.0
(1986) ELO 1676 - 30 secs/move
ZX Spectrum 48K / Cyrus Chess II
(1986) ELO 1525 - Level 4
Commodore 64 / Chessmaster 2000
(1986) ELO 1621 - Level 3
Atari 800XL / Sargon III
(1985) ELO 1569 - Level 3

The tournament was held on the basis of a calendar of round-trip matches, called "round-robin", so that each software/system had the same number of matches with the White pieces. All the matches of the tournament have been collected and are available on request in PGN format.

In the final ranking the winner is Cyrus II/ZX Spectrum with 3.5 points, 1st place together with Sargon III/Atari 800XL who scored the same on the board, but Cyrus II prevailed for "tie-break" given the highest number of wins (three) obtained with the Black pieces. All other criteria to impose a tie-break and therefore a winner were in total balance. In case of a round-robin tournament, in fact, to break the tie and have a clear winner, a number of comparisons are usually performed on the tournament results, starting from the results of the head-to-head games, and then using the Sonneborg-Berger method (sum of the points obtained by the opponents beaten and

Program	Sargon 3 Atari 800XL	Cyrus 2 ZX Spectrum	Chessmaster 2000 C64	Colossus 4.0 Amstrad CPC464	White / Black	Overall	Rank
Sargon 3 Atari 800XL	X	0-1	1-0	1-0	2	3.5	1
		0-1	1/2-1/2	1-0	1.5		
Cyrus 2 ZX Spectrum	0-1	X	0-1	1/2-1/2	0.5	3.5	1
	0-1		0-1	0-1	3		
Chessmaster 2000 C64	1/2-1/2	0-1	X	1-0	1.5	2.5	2
	1-0	0-1		1-0	1		
Colossus 4.0 Amstrad CPC464	1-0	0-1	1-0	X	2	2.5	3
	1-0	1/2-1/2	1-0		0.5		

Figure 1 - The tournament's table with the final standings

half of the points obtained by the opponents the player has drawn with), then to a greater weight of the wins and, finally, to the results with the Black pieces.

The virtual award as "best match of the tournament" goes to the game played between ChessMaster 2000 and Colossus 4.0 in the fourth round, first match, ended with the victory of the White.

I now leave the floor to Mathias who will describe in detail how the tournament was arranged and, above all, how we achieved to the final result, which in some ways surprised both of us.

The Tournament

For this 1st RMW Home Computer 8-bit Chess Tournament, generally every 8-bit system and every chess program could be considered. In doing so, the choice of chess programs is even much larger than that of the home computers of the time. The very popular systems such as C64, Atari 800XL and Amstrad CPC 464 could already be equipped with a floppy disk drive, whereas the ZX Spectrum was still loaded via a tape drive. Apart from that, these computers provided a suitable coherent framework for the first cross-platform commercial chess programs. Especially for amateur chess players, the market also offered a whole bunch of table chess computers, which were offered with specifically designed hardware inside of elaborately crafted wooden cases. These also served as playing board. Until that time the computer had to adapt to the habits of humans to play on a wooden board with real pieces. Nowadays it's the other way around and humans have adapted to the computer playing online chess by clicking with the mouse at the screen. From this point of view the 8-bit computers were the pioneers to the common and widespread way of using computers for playing chess today. Back then the programs also tried to give the human player the same view of the board as if the player himself was sitting in front of it. These mostly graphically elaborate 3D perspectives were also often an important sales argument for the purchase of such a program. The less impressive 2D representation was more familiar from books and had not yet arrived in the Zeitgeist.

Together with the mentioned representatives of the 8-bit home computers, the participating chess programs should be among the favourites according to the gaming magazines and the numerous computer vs. computer and computer vs. human tournaments. Furthermore, the initial publication of the actual programs should occur in the same year. Because continuous development would certainly result in stronger programs and would have distorted the score. According to that the chosen programs Colossus 4.0, Chessmaster 2000 and Cyrus 2 are actually from the same year 1986, only Sargon 3 is one year older than its competitors. The arrangement of the programs to the related host machine was decided quite pragmatically by lot. Alternatively, the limitation to one of these chess programs would be conceivable, which then plays matches versus itself on different host systems. This attempt would have provided a statement about the best hardware-software combination. But the focus is on determining a certain dominance of one of the tested chess programs not taking account of the used platform. Just after actually finding such a dominance, a second run could have detected the optimal hardware environment.

The programs are equipped with a comparable opening library of about 50 - 100 thousand moves. Therefore, the openings were handled correctly in the games, as far as one can tell according to the current state of theory. Early serious positional errors or even blunders were not to be found, although no program would have provoked this with targeted opening traps. The tight time limit was chosen to obtain quite quick results. The guideline is 30 seconds per move or a corresponding level setting. This adjustment leads to games with a length from 1 to 2 hours. After 12 games of round robin, the field is still close together and an outstanding dominance cannot be confirmed by any of the programs, although this is often conveyed in the self-promotion of the products. To name a clear favourite, many more matches would have had to be played. In this respect, only a snapshot is available,

but the course of the games also provides indications for comparative observation. Similar to a match between human opponents, both sides sometimes make dubious moves that involves big fluctuation in the course of the games.

Such a mediocre move was often answered with a likewise mediocre move that totally ignored the actual weakness of that move. Only when it comes to definite bad positional planning decisive counter-attack emerges. Although no major mistakes were made, the basically solid gameplay of the engines was partially overridden under time pressure. Cyrus 2 is the exception. As a shared winner of the tournament, the fast way of playing in the middlegame was particularly impressive. The 30 seconds per move was undercut by the program almost throughout with better moves than the competitors despite using the full time. The program is recommended for impatient casual players who are looking for a strong opponent.

But the two match leaders also reveal at various situations that basic and well-known chess principles have not been taken-into account. For example, the isolation of a pawn is favourable for the opponent, because without being covered by another pawn it could be attacked easily. In the first game of round 6 (21st move) Chessmaster 2000 did not handle the exchange in the appropriate way. Just as it is incomprehensible that Cyrus refuses the draw by threefold repetition in the 74th move of the same game just to go for a win in a completely lost position. And even rookies know more effective ways to checkmate the king in the endgame with a rook as was performed by Chessmaster 2000 in this game. In this way the program approaches the checkmate with apparently stark and stiff algorithms. What is special about the top chess program titles was the cross-platform availability. The systems were at least very similar in terms of bus width of 8-bit and the size of the physical RAM. As mentioned above, the data storage medium also played an important role for the consumer. The limitation of the storage capacity by using a tape cassette even had the bad consequence that a different and weaker engine was built-in as in the case of the tape-version of Chessmaster 2100 (1).

Figure 2 - Cyrus II - CM 2000 move 21

In the same manner the clock frequency of a Z80 processor with 3.5 MHz for the ZX Spectrum or 4 MHz for the Amstrad CPC is clearly above that approximate 1 MHz clocked frequency in the 6510 processor of the C64. At least theoretically, this could be utilized for a more powerful gameplay on these devices. And in fact, the processors of the table chess computers of this period have clock frequencies in the range of 5 MHz, which leads to a comparable playing strength as the strongest program in the tournament, Colossus 4.0 on Amstrad, in terms of the ELO rating (2).

However, the following consideration is important in this context. The search depth of an evaluation for a certain position is usually given in steps of a half-move - a so-called ply. This depth indicates vividly speaking how many moves the program calculates in advance. In search for the optimal move, the program must simulate as many different eligible positions as possible. A fast search algorithm is helpful in calculating these potential positions. The speed is specified in Nodes Per Second - NPS for short. Visual wise, a program with high NPS has a much finer-branched search tree. By calculating faster, the processor provides a broader base for further evaluations. A process that obviously gains in importance with an increasing number of plies.

Surprisingly, the practice of programming modern chess engines has made clear that the influence on the actual playing strength is rather marginal (3). On the one hand, there is a program that can examine many positions with high NPS in a rather inaccurate way. On the other hand, there is a program that analyses fewer positions with low NPS in a very exact way. The programs can differ up to a factor of 10 in the NPS number and still play in the same league. A high clock frequency has an advantage for the NPS number, because the computationally intensive search algorithm can be executed faster. In retrospect, this fact alone is not enough to make the program automatically better.

Nowadays, this small tournament would certainly not taken very seriously by chess engine programmers regarding a statement about the concrete playing strength of the participants. Since there are for quite a while specialized tools, to test the particular abilities of an engine. They help to let the engines compete against each other thousands of times and gave a very good prediction up to 1 ELO exact (4).

Another method of testing the quality of an engine is to specify a certain chess position and pass it on to the program to determine the next move. Thereby can be fairly

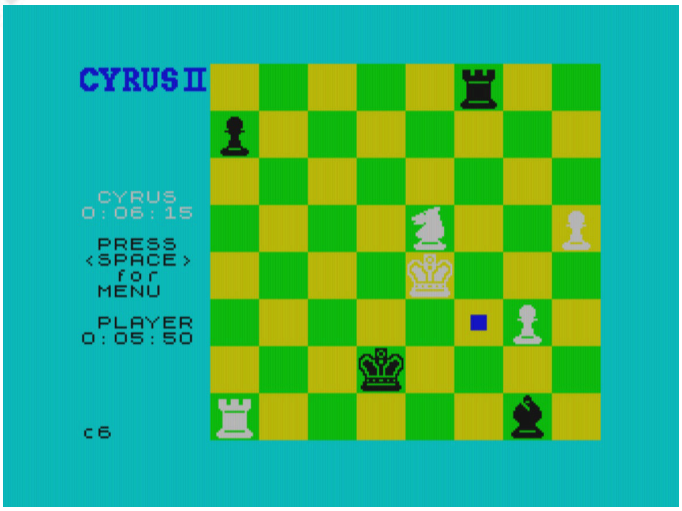


Figure 3 - Cyrus II - CM 2000 move 74

objectively determined whether the correct move is found at all and whether it is available in a reasonably short period of time.

In the eighties, this kind of automation was still quite far away. Top players tended to smile at the chess programs on home computers. But therefore, the affordable programs were equipped with thick, comprehensive manuals. These explained the rules of chess to the willing beginner and even the experienced player could learn by means of example games.

Moreover, even entire historical essays on chess were passed on to the public in this way. As soon as the interest was stirred up, the programs nevertheless presented the casual player a serious opponent. Hopefully a little bit of this pioneering spirit could be revived during this tournament. At the end there will be a few brief statements about the individual games in the manner of a sports reporter to give the reader a clue about the course of the matches.

1st Round

1st game: Colossus 4.0-Chessmaster 2000 | 1-0
Colossus leaves the middle game not unscathed, but finds his way back into the game and wins with a nice mate.
2nd game: Sargon 3-Cyrus 2 | 0-1
Cyrus was given a pawn as a present in the opening just to conquer one more figure in the end.

2nd Round

1st game: Chessmaster 2000-Sargon 3 | 1/2-1/2
A victory by draw for Chessmaster. Sargon showed very weak endgame treatment despite superior position and one pawn more.
2nd game: Cyrus 2-Colossus 4.0 | 1/2-1/2
Colossus had a better position but dissipated his advantage with an unnecessary queen manoeuvre. In the end he rescues himself with a draw.

3rd Round

1st game: Chessmaster 2000-Cyrus 2 | 0-1
After a serious positional error in the opening, Cyrus wins a figure and then the whole game.
2nd game: Colossus 4.0-Sargon 3 | 1-0
The opening actually offered chances because Colossus did not play optimally. But in the end Sargon showed suicidal tendencies.

4th Round

1st game: Chessmaster 2000-Colossus 4.0 | 1-0
A very entertaining game with opportunities on both sides. In the endgame an extra pawn decided in favour of Chessmaster.
2nd game: Cyrus 2-Sargon 3 | 0-1
The surprise of the tournament. Sargon wins his first game against previously unbeaten Cyrus 2, who gives away an advantage and was overrun after inaccurate moves.

5th Round

1st game: Sargon 3-Chessmaster 2000 | 1-0
After a risky attack, Sargon takes almost 30 moves to find the mate in the endgame with a pair of bishops and a rook versus only a rook.
2nd game: Colossus 4.0-Cyrus 2 | 0-1
Colossus slowly gains a quality just to run unnecessarily into a counterattack, which finally wipes him out.

6th Round

1st game: Cyrus 2-Chessmaster 2000 | 0-1
One move before the threefold repetition Cyrus gives away the chance for a safe draw and loses an important game.
2nd game: Sargon 3-Colossus 4.0 | 1-0
A thrilling fight that only becomes apparent in the initially perfectly balanced endgame after Colossus committed a serious positional error.

References

(1) http://www.spacious-mind.com/html/commodore_c64_fidelity_chessma.html
(2) https://www.schach-computer.info/wiki/index.php/Wiki-Elo-Liste
(3) https://www.stmintz.com/ccc/index.php?id=198960
(4) http://rebel13.nl/misc/nice.html

(R1) Alphazero Vs Stockfish – 100 games match https://www.chess.com/news/view/google-s-alphazero-destroys-stockfish-in-100-game-match

(R2) AlphaZero Vs Stockfish – 1000 games match https://www.chess.com/news/view/updated-alphazero-crushes-stockfish-in-new-1-000-game-match

An introduction to ARexx – part 1

by Gianluca Girelli

This article first appeared on Bitplane pages in November 2011.

ARexx is a scripting language born as a direct implementation of the REXX language on Amiga. It includes a number of features unique to Amiga that extend its functionality beyond the standard REXX.

Like most REXX implementations, ARexx is an interpreted language. Programs written for ARexx are called "scripts" or "macros" and several software still offer the ability to run ARexx scripts from their main interface (e.g. Personal Paint) or in their code (e.g. Hollywood). ARexx can easily communicate with all third-party software that implements a "port" for the exchange of data through a system called "IPC" (Inter Process Communication).

1. A little history

The first appearance of ARexx programming language dates back to 1987, when William S. Hawes ported for Amiga from the IBM world.
ARexx is based on the REXX (REstructured eXtended eXecutor) language described by Mike Cowlishaw in his book "The REXX Language: A Practical Approach to Programming". ARexx was included by Commodore in AmigaOS2.0 in 1990. This latest version of ARexx follows the REXX "official" language very faithfully, so much so that Hawes was later included in the working group that laid the "ANSI standard" foundations of REXX.

ARexx is written in Assembly for M68000, so its performance is not optimized to work at the maximum speed allowed by the new PPC CPUs, for which a new version of the language has never been developed. William Hawes is no longer involved in language development today (probably due to past legal issues with Commodore) and no Amiga company is currently funding new versions.
It is also not possible to extend the language by users, as for example happens in the GNU/Linux environment, since the source code is not available.
The future of this type of language therefore probably resides today in Python (already present on Amiga) but, pending an official successor, ARexx is still being used and still has a lot to say. It is emblematic that, as reported on the official AmigaPython website (see bibliography), this language supports ARexx so that it can then be replaced directly during the scripting phase. According to the website, Python is inferior than Arexx because the required memory consumption is 10 times higher, although, it must be said, this factor is no longer a problem on today's computers.

2. Why ARexx

ARexx can send commands to functions and to different applications in a single script, thus offering the opportunity to combine functions belonging to different programs.
For example, an ARexx script can extract data from a database, insert it into a spreadsheet to process it, retrieve the resulting tables and graphs from it and page them with a word processor, and then email the resulting document to your contacts in the address book. If all this looks familiar you are not wrong: Microsoft Visual Basic for Applications (VBA) has been doing it for years. Once again, however, we can say: "Amiga used to do it before!".

Despite being closed source software, the REXX language has been consolidated and expanded enormously thanks to an extremely active and competent user community. That's why ARexx still doesn't feel the weight of the years. Forged around real work/use needs of the "community", it incorporates an extremely powerful and flexible set of instructions that can work efficiently and effectively on input/output formatting (especially in the field of strings) allowing extreme ease of reading/data transmission.

It is also because of this type of capability that Commodore decided, starting with AmigaDOS Release 2, to insert ARexx into the operating system. This, as a result, excluded AmigaBASIC that proved slow, slapdash and, above all, unable to make the most out of Amiga's innate multitasking capabilities.

Like all REXX implementations, ARexx uses a typeless representation of data, meaning it makes no distinction between integers, real (floating-point) numbers, strings, characters, vectors, etc. All data is represented internally as character strings, making it easier to write expressions and algorithms. As is often the case in dynamic languages, variables do not need to be declared before they can be used, but are created when they are first used.

The ARexx command set is simple, but in addition to the

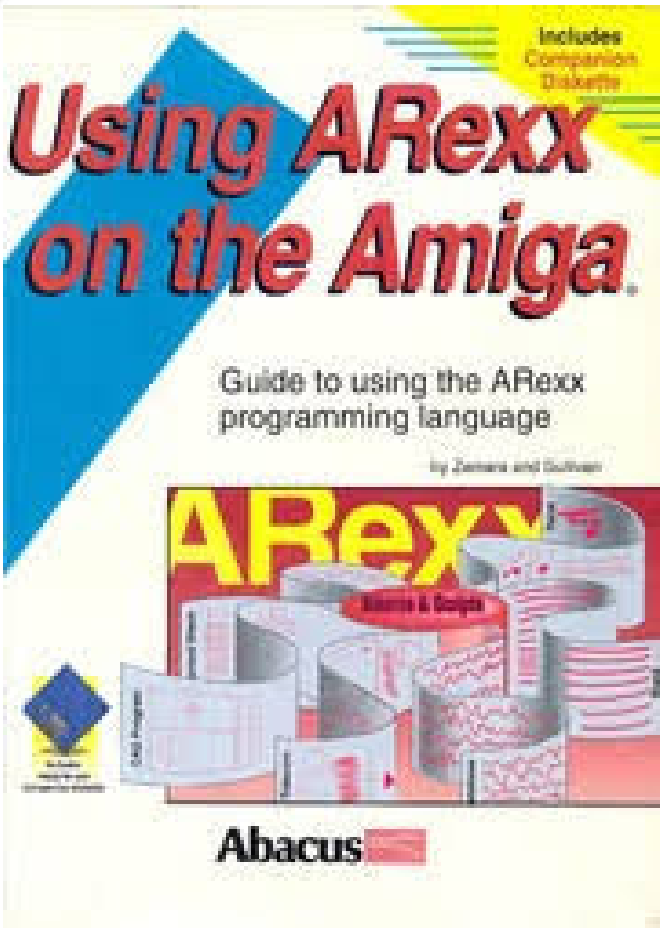


Figure 1 - Using ARexx on the Amiga

language commands there are the functions included in the relative Amiga library (rexxsyslib.library) in "System:Libs/". It is also easy to add other libraries or functions because ARexx scripts can be invoked as functions by other scripts. Any program for Amiga that incorporates an "ARexx port" can share its functions with ARexx programs.

3. A little practice

Instead of starting with the classic "Hello World!", let's try a simple program that calculates age in days.

```
/* age.rexx */
say 'write age'
pull age
say 'you have' age*365 'days'
```

As simple as it may be, this code snippet tells us that:
- all ARexx programs must start with a comment (the sequence /* */). The actual content of the comment does not matter;
- variables (in this case "age") need not to be explicitly declared as they are dynamically allocated;
- the concatenation of the output strings (commanded by "say") is performed by the character "space" (or blank) unlike other languages that use well-defined operators.

Arexx is a case-insensitive language. It is worth noting, however, that the "pull" command used to read the input is the abbreviated form of "parse upper pull". Each alphanumeric string read with this method will always be returned in uppercase letters, i.e. the input strings "Amiga" or "AmIgA" (or any other combination) will always be returned by "say" as: "AMIGA".

To use the program just open your favorite text editor, type the code and save it (in text mode) as file_name.rexx (example: age.rexx). At this point just open the shell and type ">rx age.rexx" (or use the short form ">rx age") and the trick is done. The system will automatically invoke the ARexx interpreter "RexxMast" and execute the script. Of course, if the shell is opened in a directory other than the one that contains your code, you will need to type the full path.

Let's now move on to the infamous "Hello World!", which in ARexx is simply:

```
say "Hello World"!
```
(from command line)

or

```
/* hello.rexx */
say "Hello World!"
```
(from script)

As you can see, the use of double quotation marks is an alternative to single quotation marks and is useful when we need to use accents or apostrophes within a string. As an example, if you were to write that code in Italian where age is translated as eta', the following form "eta' " is less prone to error than writing 'eta''.

ARexx can open windows that are independent from the shell we are working in. Here's an example.

```
/* ARexx base constructs test */
clear
   if~open('console','con:0/0/640/200/
RexxWindow/CLOSE','W')
   then exit 20
  call writeln'console','Hello everybody.
This is a windows usage test.'
  call writeln'console','Press RETURN to
continue.'
  call readln'console'
  call close 'console'
```

```
end
```

In this code we first open a window (parameter 'console') of 640x200 pixels, called RexxWindow equipped with closing gadgets.
If for some reason the window cannot be opened, execution is stopped by returning an error message.
'Console' represents a logical file to which input/output is redirected through the "writeln" and "readln" commands, invoked by the "call" statement. The window is then closed (call close 'console') and the program terminated (end). Be careful not to confuse "end", which usually closes a block of code with the "exit" statement which, if used, immediately terminates the script from any point it is invoked.

We now use the same type of system to work on physical files. The following code creates a simple text file on the logical disk "Ram" (prefix.dat) containing telephone prefixes (area codes):

```
/* writes area codes in  prefix.dat */
datafile='datafile'
filename='ram:prefix.dat'
if open(datafile, filename, 'w') then do
   call writeln(datafile, '02 Milan')
   call writeln(datafile, '06 Rome')
   call writeln(datafile, '045 Verona')
   call writeln(datafile, '081 Naples')
   call close(datafile)
   end
else
   say "Unable to open " filename
```

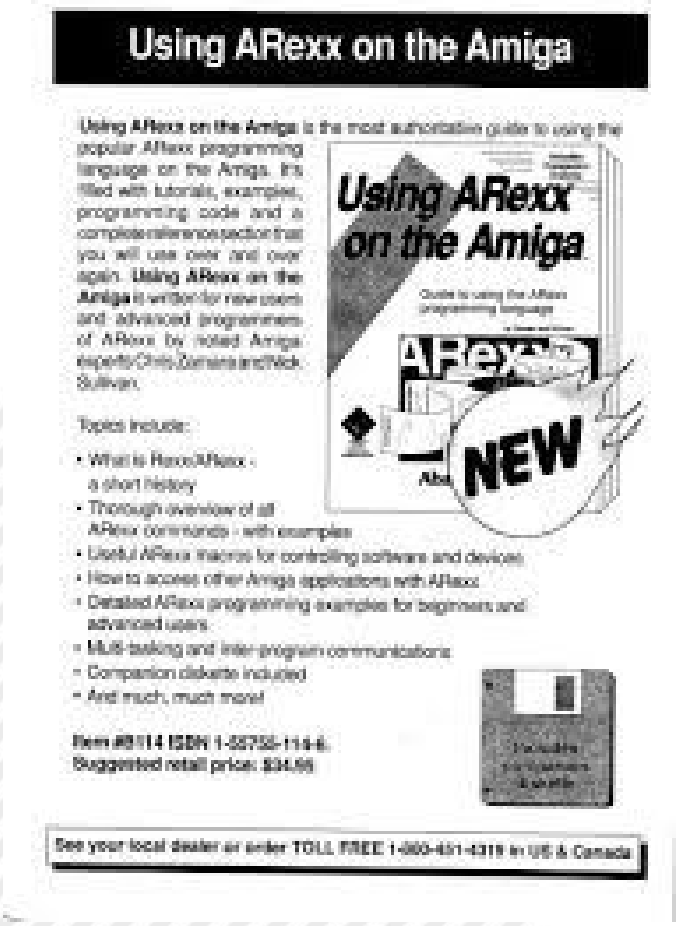


Figure 2 - Using ARexx on the Amiga advertisement

Being this a text file, it can be read by simply typing the command from the shell

```
Shell> type ram:prefix.dat
```

If we want to read the file from inside another script, just use:

```
/* reads the records inside prefix.dat */
tfile='typefile'
name ='ram:prefix.dat'
if open(tfile, name, 'r') then do
  do while ~eof(tfile)
     say readln(tfile)
     end
  call close(tfile)
  end
else
   say 'Unable to open file' name '.'
```

This script, which does not differ from the previous "type" command, can, however, be modified as desired to load the data read from the file into variables (for example, a text array) so that it can then be processed as needed.

4. Conclusions

As we have seen in this brief introduction Arexx, despite the weight of the years, is still a versatile and modern language able to compete with more renowned products. At the time this article was originally written, as far as the Amiga world was concerned, ARexx was (and is) present on every version of the operating system from 2.0 onwards; ANSI compliant REXX implementations still exist for all platforms (Unix/Linux, Mac, Windows etc.) and the most popular of these was undoubtedly Regina_REXX, which became AROS's basic ARexx. MorphOS, on the other hand, having a rexxsyslib.library that for various reasons did not work, was unable to use ARexx except by replacing the native MorphOS library with one from the AmigaOS 3.x family. The Regina_REXX project therefore also set itself the goal of becoming the standard REXX for MorphOS, thus freeing itself from the system's dependence on the AmigaOS 3.x

library.
The ease with which you write scripts in ARexx is such that you don't have to be a programmer to implement them.
However, the power of ARexx (and its ability to interface with almost every Amiga application both modern and "legacy") makes this language essential and useable even by the most experienced programmer, especially for those routine jobs for which processing speed is not an important factor.

5. Next issue preview

In the next tutorial we will delve into the use of strings, arrays, conditional instructions and procedures. At the end of the tutorial we will have laid the foundations for the construction of a simple game engine for text adventures.

6 Bibliography

Mike Cowlishaw "The REXX Language: A Practical Approach to Programming" (1985) Prentice Hall. ISBN 0-13-780651-5.

Chris Zamara, Nick Sullivan "Using Arexx on the Amiga" (1991) Abacus Software Inc. ISBN 1-55755-114-6.

AmigaOS 2.0 System Manual

Figure 3 - The REXX Language

Scripting languages
In computer science, a scripting language is an interpreted programming language, usually intended for the automation of the operating system (batch) or applications (macro), or for use within web pages.
Scripts are generally simple programs whose purpose is to interact with other programs, much more complex, in which the most significant operations take place. Scripts are distinguished from the programs with which they interact, usually implemented in a different and often non-interpreted language. In addition, scripts are often created or edited by the end user.

Interpreted languages
Languages that involve the execution of code lists through an interpreter are called interpreted languages.
An interpreter has the purpose of running a program in a high-level language, without first compiling it.
In fact, it has the task of executing the instructions in the language used, translating them from time to time into instructions in machine language.

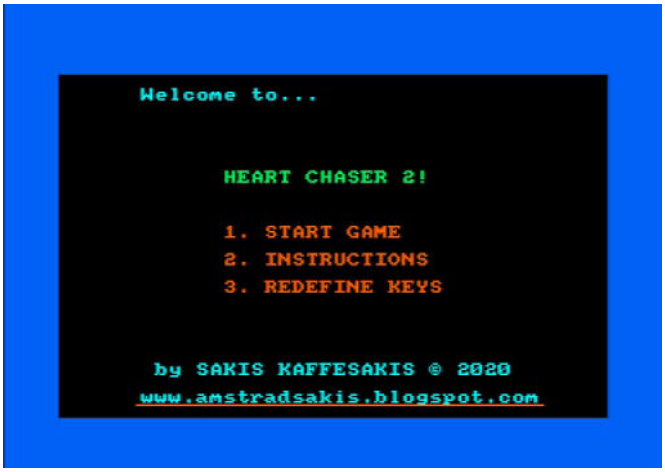
Legacy programs
In computer science, the term "legacy" refers to all those data and applications that have been inherited from languages, platforms and programming techniques developed before current technology. Most companies that use computers extensively have applications and databases of this type that support critical areas of service. Typically the challenge is to keep legacy code in use while you are migrating to newer, more efficient code that uses new technologies and increased programmer capabilities.

From the ARexx manual
"ARexx was developed on an Amiga 1000 computer with 512k bytes of memory and two floppy disk drives. The language prototype was developed in C using Lattice C, and the production version was written in assembly-language using the Metacomco assembler. The documentation was created using the TxEd editor, and was set in TeX using AmigaTeX. This is a 100% Amiga product."

HEART CHASER 2 - a Locomotive Basic game

by Sakis Kaffesakis - introduction by Francesco Fiorentini

I didn't have time to finish writing the closing article of the previous issue, in which I invited readers to participate with their work, that on the RetroMagazine page we were contacted by Sakis Kafesakis, manager of the blog: https://amstradsakis.blogspot.com/.
Sakis invited us to visit his website and take a look at his works in Locomotive Basic. The opportunity was too tempting not to take advantage of it, so we tried to ask him if he was interested in sharing the code of one of his games with us, to the advantage of all the readers of RetroMagazine World.
Well, his answer wasn't long in coming and below you can find the code and explanations of one of his latest games: Heart Chaser 2.



Heart Chaser 2 is the sequel to Heart Chaser 1 (who guessed it?) with new features and advanced levels!In this game you have to collect hearts, without being captured by the increasingly fierce enemies.
In each of the 7 levels you have to collect 6 hearts, paying attention to the enemies that become more and more risky increasing the level.
Watch out, the level can be completed if you are not captured, but to move on to the next level you must collect the 6 hearts.
You only have 3 lives at your disposal to complete the task. (Shh, there is a secret passage in the menu to get 10 lives...).

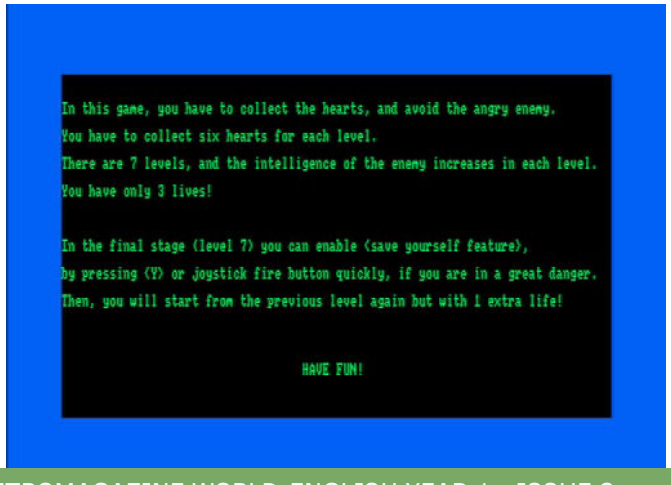
Ecco il link diretto per scaricare il gioco:
https://drive.google.com/drive/folders/1hZcn2TQ0GRmKUW4oY4IGR-eiDlEpcan8
Ed un longplay dello stesso:
https://www.youtube.com/watch?v=vxcUiwp18hM&feature=emb_title

Game structure:

```
10-150 INTRO-MENU
150-900 MAIN GAME
1600-1690 DEFINING OPPONENTS MOVE
1900-1999 DEFINING INITIAL OPPONENT POSITION IN EVERY STAGE
2000-3080 INSTRUCTIONS- REDEFINE KEYS
4000-4610 STAGES DESIGN
4700-4830 LIMITING HEART APPEARANCE POINT DEPENDING ON STAGE
5000-5720 LIMITING PLAYER MOVE DEPENDING ON STAGE
6000-6720 LIMITING OPPONENT 1 MOVE DEPENDING ON STAGE
7000-7720 LIMITING OPPONENT 2 MOVE DEPENDING ON STAGE
```

X,Y = player's position.
OX, OY = previous player position (if players choose to go on a blocked point or outside the map, these variables are used to force him stay at the previous point).

```
320 a$=INKEY$
330 IF A$=PLA$ AND X>1 AND X<40 THEN OX=X:OY=Y:X=X-1:GOTO 390
340 IF A$=PLD$ AND X>1 AND X<40 THEN OX=X:OY=Y:X=X+1:GOTO 390
350 IF A$=PLK$ AND Y>2 AND Y<25 THEN OX=X:OY=Y:Y=Y+1:GOTO 390
360 IF A$=PLP$ AND Y>2 AND Y<25 THEN OX=X:OY=Y:Y=Y-1:GOTO 390
365 IF DIFF=10 AND (A$="Y" OR A$="y" OR A$="X" OR A$="Z") THEN GOTO 8000
370 GOTO 400
390 IF X=1 THEN X=2
```



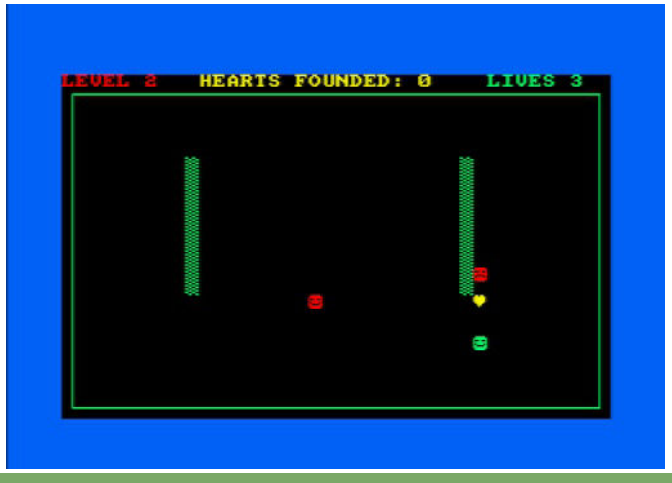
```
391 IF X=40 THEN X=39
392 IF Y=2 THEN Y=3
393 IF Y=25 THEN Y=24
395 GOSUB 5000
399 LOCATE OX,OY:PRINT FIGK$
400 LOCATE X,Y:PRINT FIG1$
```

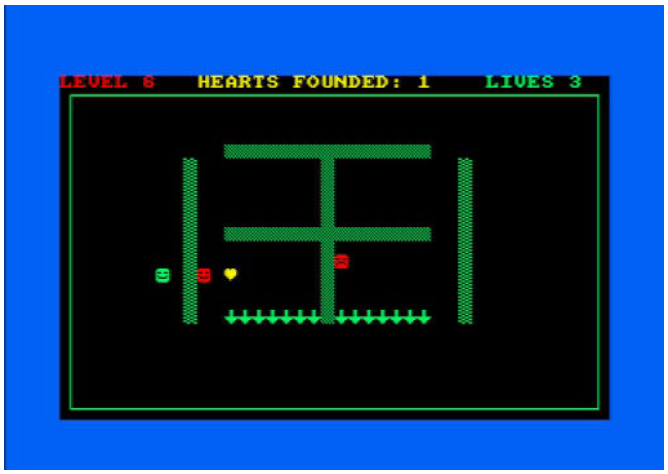
390-393 are standard limits of all stages, in 5000 there are the limits of each stage.
399 prints blanc on the previous point and 400 prints the player in the new position.
400 show the player in the new position.

The same way is used for opponents move: using XZ1,YZ1 for opponent 1 variables.
Variable ANT shows the decision of the opponent 1 for where to move.

```
510 IF ANT=1 THEN OXZ1=XZ1:OYZ1=YZ1:XZ1=XZ1-
1:GOTO 550
515 IF ANT=2 THEN
OXZ1=XZ1:OYZ1=YZ1:XZ1=XZ1+1:GOTO 550
520 IF ANT=3 THEN OXZ1=XZ1:OYZ1=YZ1:YZ1=YZ1-
1:GOTO 550
525 IF ANT=4 THEN
OXZ1=XZ1:OYZ1=YZ1:YZ1=YZ1+1:GOTO 550
550 IF XZ1=1 THEN XZ1=2
560 IF XZ1=40 THEN XZ1=39
561 IF YZ1=2 THEN YZ1=3
562 IF YZ1=24 THEN YZ1=23
563 GOSUB 6000
580 LOCATE OXZ1,OYZ1:PRINT " "
590 PEN 2:LOCATE XZ1,YZ1:PRINT FIG8$:PEN 1
```

Opponents intelligence:
Variable DIFF is set on 4 in the start of the game and increases by 1 at each stage. It makes the enemy getting smarter, as we get to next stages.





Variable ANT shows the decision of the opponent 1 for where to move.
It is set randomly on 500, but goes for further check (GOSUB 1600).

500 ANT=INT(RND*4)+1:GOSUB 1600

By using 1610 and 1615, for example at stage 1 where DIFF=4,there is a high possibilty of <RETURN> at 1615 and maintain the random move.

In next stages, where DIFF is increased (eg 10), it is almost sure that the random move will not occur.

So, as we get to further stages, the possibility of the random move is limited.

1620-1623 decide the movement of the opponent when they have the same one same coordinate (x or y).

Then, 1630 decides by random what the correction will be, if player and opponent have different both coordinates. So, if player and opponent have different x and y, there is 50% for the opponent to correct X variable and 50% to correct y variable.

```
1610 ANT1=INT(RND*10)+1
1615 IF ANT1>DIFF THEN RETURN
1620 IF X=XZ1 AND Y<YZ1 THEN ANT=3:RETURN
1621 IF X=XZ1 AND Y>YZ1 THEN ANT=4:RETURN
1622 IF Y=YZ1 AND X<XZ1 THEN ANT=1:RETURN
1623 IF Y=YZ1 AND X>XZ1 THEN ANT=2:RETURN
1630 KAT=INT(RND*2)+1
1635 IF KAT=1 THEN GOTO 1650
1637 IF KAT=2 THEN GOTO 1654
1650 IF X<XZ1 THEN ANT=1
1652 IF X>XZ1 THEN ANT=2
1653 GOTO 1669
1654 IF Y<YZ1 THEN ANT=3
```

```
1656 IF Y>YZ1 THEN ANT=4
```

Last notes: to ensure that, in the final advanced stage, the 2 opponents will not make exactly the same movement decisions, we made the 1st opponent not to be able to move in the bottom line (the sad face) and the 2nd opponent (the smile face) to have a corrected small possibility of a random move.
This, sometimes, results in the player to be surrounded by the enemies.

Here is the full code of the game in Locomotive Basic:

```
10 REM HEART CHASER 2! COPYRIGHT BY SAKIS KAFFESAKIS 2020.
15 GOSUB 3100
16 GOSUB 3600
20 CLS:MODE 1:BORDER 11:PEN 3:INK 3,20:INK 0,0:RANDOMIZE TIME:SCORE=0:DIFF=4:LIVES=3
25 IF KRYFO=1 THEN LIVES=10
30 LOCATE 7,2:PRINT "Welcome to..."
40 LOCATE 13,8:PEN 2:SPEED INK 40,40:INK 2,19,24:PRINT "HEART CHASER 2!"
50 PEN 1:INK 1,15:LOCATE 13,12:PRINT"1. START GAME":LOCATE 13,14:PRINT"2.
INSTRUCTIONS":LOCATE 13,16:PRINT"3. REDEFINE KEYS"
60 LOCATE 8,22:PEN 3:PRINT"by SAKIS KAFFESAKIS":LOCATE 28,22:PRINT CHR$(164):LOCATE
30,22:PRINT"2020":LOCATE 7,24:PRINT"www.amstradsakis.blogspot.com":MOVE 90,15:DRAW 565,15
70 A$=INKEY$
80 IF A$="2" THEN GOSUB 2000:GOTO 20
90 IF A$="3" THEN GOSUB 3000:GOTO 20
100 IF A$="1" THEN GOTO 150
110 IF A$="L" OR A$="l" THEN GOSUB 8100:GOTO 30
140 GOTO 70
150 REM START GAME
160 REM START GAME
180 MODE 1:PEN 1:INK 1,19:
190 GOSUB 1000:GOSUB 1900
200 MOVE 13,13:DRAW 626,13:DRAW 626,376:DRAW 13,376:DRAW 13,13
250 LOCATE X,Y:PRINT FIG1$
260 PEN 2:INK 2,6:LOCATE XZ1,YZ1:PRINT FIG8$:LOCATE XZ2,YZ2:PRINT FIG9$:PEN 1
263 REM FOR DIFFK=1 TO 5:DIFF=2*DIFF
265 REM FOR ST=1 TO 2
280 PISTA=DIFF-3
285 ON PISTA GOSUB 4000,4100,4200,4300,4400,4500,4600
290 ON PISTA GOSUB 4700,4720,4740,4760,4780,4800,4820:REM
OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3
300 WHILE 1
301 LO=0:PISTA=DIFF-3
302 LOCATE 1,1:PEN 2:PRINT "LEVEL";DIFF-3:PEN 3:LOCATE 11,1:PRINT"HEARTS
FOUNDED:";SCORE:LOCATE 32,1:PEN 1:PRINT"LIVES";LIVES;
305 PEN 3:INK 3,24:LOCATE OBJX,OBJY:PRINT OBJ$:PEN 1
320 a$=INKEY$
330 IF A$=PLA$ AND X>1 AND X<40 THEN OX=X:OY=Y:X=X-1:GOTO 390
340 IF A$=PLD$ AND X>1 AND X<40 THEN OX=X:OY=Y:X=X+1:GOTO 390
350 IF A$=PLK$ AND Y>2 AND Y<25 THEN OX=X:OY=Y:Y=Y+1:GOTO 390
360 IF A$=PLP$ AND Y>2 AND Y<25 THEN OX=X:OY=Y:Y=Y-1:GOTO 390
365 IF DIFF=10 AND (A$="Y" OR A$="y" OR A$="X" OR A$="Z") THEN GOTO 8000
370 GOTO 400
390 IF X=1 THEN X=2
391 IF X=40 THEN X=39
392 IF Y=2 THEN Y=3
393 IF Y=25 THEN Y=24
395 GOSUB 5000
399 LOCATE OX,OY:PRINT FIGK$
400 LOCATE X,Y:PRINT FIG1$
430 IF X=OBJX AND Y=OBJY THEN score=SCORE+1:OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3:GOSUB
3500:ON PISTA GOSUB 4700,4720,4740,4760,4780,4800,4820
450 IF X=XZ1 AND Y=YZ1 THEN GOTO 1500
451 IF X=XZ2 AND Y=YZ2 THEN GOTO 1592
500 ANT=INT(RND*4)+1:GOSUB 1600
505 OXZ1=XZ1:OYZ1=YZ1
510 IF ANT=1 THEN OXZ1=XZ1:OYZ1=YZ1:XZ1=XZ1-1:GOTO 550
515 IF ANT=2 THEN OXZ1=XZ1:OYZ1=YZ1:XZ1=XZ1+1:GOTO 550
520 IF ANT=3 THEN OXZ1=XZ1:OYZ1=YZ1:YZ1=YZ1-1:GOTO 550
525 IF ANT=4 THEN OXZ1=XZ1:OYZ1=YZ1:YZ1=YZ1+1:GOTO 550
550 IF XZ1=1 THEN XZ1=2
560 IF XZ1=40 THEN XZ1=39
561 IF YZ1=2 THEN YZ1=3
562 IF YZ1=24 THEN YZ1=23
563 GOSUB 6000
580 LOCATE OXZ1,OYZ1:PRINT " "
```

```
590 PEN 2:LOCATE XZ1,YZ1:PRINT FIG8$:PEN 1
600 IF X=XZ1 AND Y=YZ1 THEN GOTO 1500
601 IF X=XZ2 AND Y=YZ2 THEN GOTO 1592
605 ANTIP=INT(RND*4)+1:GOSUB 1670
606 OXZ2=XZ2:OYZ2=YZ2
607 IF ANTIP=1 THEN OXZ2=XZ2:OYZ2=YZ2:XZ2=XZ2-1:GOTO 612
608 IF ANTIP=2 THEN OXZ2=XZ2:OYZ2=YZ2:XZ2=XZ2+1:GOTO 612
609 IF ANTIP=3 THEN OXZ2=XZ2:OYZ2=YZ2:YZ2=YZ2-1:GOTO 612
610 IF ANTIP=4 THEN OXZ2=XZ2:OYZ2=YZ2:YZ2=YZ2+1:GOTO 612
612 IF XZ2=1 THEN XZ2=2
613 IF XZ2=40 THEN XZ2=39
614 IF YZ2=2 THEN YZ2=3
615 IF YZ2=25 THEN YZ2=24
616 GOSUB 7000
617 LOCATE OXZ2,OYZ2:PRINT " "
618 PEN 2:LOCATE XZ2,YZ2:PRINT FIG9$:PEN 1
619 IF X=XZ1 AND Y=YZ1 THEN GOTO 1500
620 IF X=XZ2 AND Y=YZ2 THEN GOTO 1592
622 IF SCORE=6 THEN SCORE=0:DIFF=DIFF+1:LO=1
623 IF DIFF=11 THEN GOTO 950
625 IF LO=1 THEN CLS:LOCATE 17,13:PRINT "LEVEL";DIFF-3:GOSUB 1496:CLS:GOTO 150
900 WEND
950 MODE 1:LOCATE 17,10:PRINT"YOU WIN!":LOCATE 10,18:PRINT"PRESS SPACE TO RESTART":GOSUB
8200:GOSUB 1495:SCORE=0:DIFF=1
952 R$=INKEY$
954 IF R$=" " THEN GOTO 20
956 GOTO 952
998 END
999 MODE 2:PEN 1:INK 1,19:END
1000 FIG1$=CHR$(224):X=20:Y=20:FIGK$=" "
1005 INK 1,19:INK 2,6:INK 3,24
1010 FIG8$=CHR$(225):XZ1=5:YZ1=5:ekr$=CHR$(252)
1012 IF DIFF<4 THEN XZ1=5:YZ1=5
1013 IF DIFF>4 AND DIFF<7 THEN XZ1=30:YZ1=10
1014 IF DIFF>7 AND DIFF<10 THEN XZ1=20:YZ1=10
1015 IF DIFF=10 THEN XZ1=20:YZ1=18
1020  REM !!!!!!!!!!!!!SETTING DIFFICULTY!!!!!!!!!!!!!!!!!
1030 OBJ$=CHR$(228)
1040 FIG9$=CHR$(224)
1050 TU$=CHR$(207):TUK$=CHR$(241):TUX$=CHR$(206):TUD$=CHR$(127):TUM$=CHR$(237):TUU$=CHR$(229)
1490 RETURN
1495
W1$=INKEY$:W2$=INKEY$:W3$=INKEY$:W4$=INKEY$:W5$=INKEY$:W6$=INKEY$:W1$=INKEY$:W2$=INKEY$:W3$=I
NKEY$:W4$=INKEY$:W5$=INKEY$:W6$=INKEY$:RETURN
1496 R$=INKEY$
1498 IF R$=" " THEN RETURN
1499 GOTO 1496
1500 SOUND 1,800,20,15,,,15:LOCATE XZ1,YZ1:PEN 2:PRINT EKR$:PEN 1:LIVES=LIVES-1:IF LIVES=0
THEN GOTO 1700
1548 GOSUB 1495
1550 D$=INKEY$
1560 IF D$="" THEN GOTO 1550
1570 MODE 1:LOCATE 17,20:PRINT"LIFE LOST":GOSUB 1495
1580 D$=INKEY$
1590 IF D$=" " THEN GOTO 150
1591 GOTO 1580
1592 SOUND 1,800,20,15,,,15:LOCATE XZ2,YZ2:PEN 2:PRINT EKR$:PEN 1:LIVES=LIVES-1:IF LIVES=0
THEN GOTO 1700
1593 GOSUB 1495
1594 D$=INKEY$
1595 IF D$="" THEN GOTO 1594
1596 MODE 1:LOCATE 17,20:PRINT"LIFE LOST":GOSUB 1495
1597 D$=INKEY$
1598 IF D$=" " THEN GOTO 150
1599 GOTO 1597
1600 REM
1610 ANT1=INT(RND*10)+1
1615 IF ANT1>DIFF THEN RETURN
1620 IF X=XZ1 AND Y<YZ1 THEN ANT=3:RETURN
1621 IF X=XZ1 AND Y>YZ1 THEN ANT=4:RETURN
1622 IF Y=YZ1 AND X<XZ1 THEN ANT=1:RETURN
1623 IF Y=YZ1 AND X>XZ1 THEN ANT=2:RETURN
1630 KAT=INT(RND*2)+1
1635 IF KAT=1 THEN GOTO 1650
1637 IF KAT=2 THEN GOTO 1654
```

```
1650 IF X<XZ1 THEN ANT=1
1652 IF X>XZ1 THEN ANT=2
1653 GOTO 1669
1654 IF Y<YZ1 THEN ANT=3
1656 IF Y>YZ1 THEN ANT=4
1669 RETURN
1670 ANTIP1=INT(RND*10)+1:IF DIFF=10 THEN ANTIP1=ANTIP1+1
1671 IF ANTIP1>DIFF THEN RETURN
1672 IF X=XZ2 AND Y<YZ2 THEN ANTIP=3:RETURN
1673 IF X=XZ2 AND Y>YZ2 THEN ANTIP=4:RETURN
1674 IF Y=YZ2 AND X<XZ2 THEN ANTIP=1:RETURN
1675 IF Y=YZ2 AND X>XZ2 THEN ANTIP=2:RETURN
1676 KAT2=INT(RND*2)+1
1677 IF KAT2=1 THEN GOTO 1680
1678 IF KAT2=2 THEN GOTO 1683
1680 IF X<XZ2 THEN ANTIP=1
1681 IF X>XZ2 THEN ANTIP=2
1682 GOTO 1690
1683 IF Y<YZ2 THEN ANTIP=3
1684 IF Y>YZ2 THEN ANTIP=4
1690 RETURN
1700 GOSUB 1496:MODE 1:LOCATE 17,20:PRINT"GAME OVER":LOCATE 10,18:PRINT"PRESS SPACE TO
RESTART":GOSUB 8300:GOSUB 1495
1710 D$=INKEY$
1720 IF D$=" " THEN GOTO 20
1730 GOTO 1710
1826 IF OBJX=26 AND OBJY=22 THEN GOTO 4820
1900 IF DIFF=4 THEN XZ1=15:YZ1=8
1902 IF DIFF=5 THEN XZ1=31:YZ1=10
1904 IF DIFF=6 THEN XZ1=32:YZ1=10
1906 IF DIFF=7 THEN XZ1=24:YZ1=10
1908 IF DIFF=8 THEN XZ1=20:YZ1=10
1910 IF DIFF=9 THEN XZ1=29:YZ1=12
1912 IF DIFF=10 THEN XZ1=29:YZ1=15
1920 IF DIFF=4 THEN XZ2=30:YZ2=10
1922 IF DIFF=5 THEN XZ2=11:YZ2=10
1924 IF DIFF=6 THEN XZ2=8:YZ2=10
1926 IF DIFF=7 THEN XZ2=14:YZ2=10
1928 IF DIFF=8 THEN XZ2=25:YZ2=10
1930 IF DIFF=9 THEN XZ2=11:YZ2=12
1932 IF DIFF=10 THEN XZ2=12:YZ2=15
1999 RETURN
2000 MODE 2:PEN 1:INK 1,19:LOCATE 1,3:PRINT"In this game, you have to collect the hearts, and
avoid the angry enemies.":LOCATE 1,5:PRINT"You have to collect six hearts for each level."
2010 LOCATE 1,7:PRINT"There are 7 levels, and the intelligence of the enemies increases in
each level.":LOCATE 1,9:PRINT"You have only 3 lives!"
2015 LOCATE 1,13:PRINT"In the final stage (level 7) you can enable <save yourself>
feature,":LOCATE 1,15:PRINT"by pressing <Y> or joystick fire button quickly, if you are in a
great danger."
2016 LOCATE 1,17:PRINT"Then, you will start from the previous level again but with 1 extra
life!"
2020 LOCATE 36,22:PRINT"HAVE FUN!"
2030 GOSUB 1495
2040 T$=INKEY$
2050 IF T$<>"" THEN GOTO 20
2060 GOTO 2040
2999 RETURN
3000 PEN 1:INK 1,19:MODE 2:LOCATE 4,6:PRINT"1. JOYSTICK KEYS":LOCATE 4,10:PRINT"2. a w s d
KEYS":LOCATE 4,14:PRINT"3. q a o p KEYS"
3005 LOCATE 3,18:PRINT"PLS DO NOT USE CAPITAL KEYS"
3010 K$=INKEY$
3020 IF K$="1" THEN PLK$="\n":PLP$=" ":PLA$=" ":PLD$="\t":GOTO 3060
3030 IF K$="2" THEN PLK$="s":PLP$="w":PLA$="a":PLD$="d":GOTO 3060
3040 IF K$="3" THEN PLK$="a":PLP$="q":PLA$="o":PLD$="p":GOTO 3060
3050 GOTO 3010
3060 LOCATE 16,22:PRINT "SELECTION OK"
3065 L$=INKEY$
3070 IF L$="" THEN GOTO 3065
3080 RETURN
3100 PLK$="\n":PLP$=" ":PLA$=" ":PLD$="\t":RETURN
3500 REM
3510 SOUND 1,90,10,15:SOUND 1,85,7,15:SOUND 1,80,7,15:SOUND 1,75,7,15:
3599 RETURN
3600 FOR SOU=1 TO 2:SOUND 1,80,10,15:SOUND 1,67,10,15:SOUND 1,60,15,15:SOUND 1,0,5:NEXT SOU
3649 RETURN
```

```
4000 FOR BL=1 TO 10:LOCATE 15+BL,10:PRINT TU$:LOCATE 15+BL,17:PRINT TU$:NEXT BL:RETURN
4100 FOR BL=1 TO 10:LOCATE 10,6+BL:PRINT TUD$:LOCATE 30,6+BL:PRINT TUD$:NEXT BL:RETURN
4200 FOR BL=1 TO 15:LOCATE 12+BL,6:PRINT TU$:LOCATE 12+BL,12:PRINT TUD$:LOCATE 12+BL,18:PRINT
TUX$:NEXT BL:RETURN
4300 FOR BL=1 TO 12:LOCATE 10,6+BL:PRINT TU$:LOCATE 20,6+BL:PRINT TUX$:LOCATE 30,6+BL:PRINT
TU$:NEXT BL:RETURN
4400 FOR BL=1 TO 12:LOCATE 10,5+BL:PRINT TUU$:LOCATE 30,5+BL:PRINT TUU$:NEXT BL
4401 FOR BL=1 TO 19:LOCATE 10+BL,6:PRINT TUU$:NEXT BL
4402 RETURN
4500 FOR BL=1 TO 15:LOCATE 12+BL,6:PRINT TU$:LOCATE 12+BL,12:PRINT TU$:LOCATE 12+BL,18:PRINT
TUK$:NEXT BL
4501 FOR BL=1 TO 12:LOCATE 10,6+BL:PRINT TUD$:LOCATE 20,6+BL:PRINT TU$:LOCATE 30,6+BL:PRINT
TUD$:NEXT BL
4502 RETURN
4600 FOR BL=1 TO 12:FOR BL1=1 TO 38:LOCATE 1+BL1,2+BL:PRINT TU$:NEXT BL1:NEXT BL
4601 FOR BL=1 TO 11:FOR BL1=1 TO 10:LOCATE 1+BL1,13+BL:PRINT TU$:NEXT BL1:NEXT BL:LOCATE 1,1
4602 FOR BL=1 TO 11:FOR BL1=1 TO 10:LOCATE 40-BL1,13+BL:PRINT TU$:NEXT BL1:NEXT BL:LOCATE 1,1
4603 LOCATE 15,18:PRINT TU$:LOCATE 15,22:PRINT TU$:LOCATE 26,18:PRINT TU$:LOCATE 26,22:PRINT
TU$
4604 LOCATE 6,4:PRINT"www.amstradsakis.blogspot.com"
4605 LOCATE 10,12:PRINT"YOU WILL NOT SURVIVE!!"
4606 LOCATE 4,20:PRINT"RUN":LOCATE 5,21:PRINT"OR":LOCATE 5,22:PRINT"DIE!":LOCATE
35,20:PRINT"I":LOCATE 33,21:PRINT"WILL":LOCATE 32,22:PRINT"CHASE":LOCATE 33,23:PRINT"YOU!"
4607 LOCATE 13,7:PRINT"Save <y>ourself!"
4610 RETURN
4700 REM OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3
4701 OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3
4702 IF OBJX>15 AND OBJX<26 AND (OBJY=10 OR OBJY=17) THEN GOTO 4701
4703 RETURN
4719 RETURN
4720 OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3
4721 IF (OBJX=10 OR OBJX=30) AND OBJY>6 AND OBJY<17 THEN GOTO 4720
4722 RETURN
4740 OBJX=INT(RND*27)+7:OBJY=INT(RND*13)+3
4741 IF OBJX>12 AND OBJX<28 AND (OBJY=6 OR OBJY=12 OR OBJY=18) THEN GOTO 4740
4742 RETURN
4760 OBJX=INT(RND*27)+7:OBJY=INT(RND*13)+3
4761 IF (OBJX=10 OR OBJX=20 OR OBJX=30) AND OBJY>6 AND OBJY<19 THEN GOTO 4760
4762 RETURN
4780 OBJX=INT(RND*27)+7:OBJY=INT(RND*13)+7
4781 IF (OBJX=10 OR OBJX=30) AND OBJY>5 AND OBJY<18  THEN GOTO 4780
4782 IF OBJX>9 AND OBJX<31 AND OBJY=6 THEN GOTO 4780
4783 RETURN
4800 OBJX=INT(RND*24)+8:OBJY=INT(RND*13)+7
4801 IF (OBJX=10 OR OBJX=20 OR OBJX=30) AND OBJY>6 AND OBJY<19 THEN GOTO 4800
4802 IF OBJX>12 AND OBJX<28 AND (OBJY=6 OR OBJY=12 OR OBJY=18) THEN GOTO 4800
4805 RETURN
4820 OBJX=INT(RND*37)+2:OBJY=INT(RND*21)+3
4821 IF OBJX<12 OR OBJX>29 THEN GOTO 4820
4822 IF OBJY<15 THEN GOTO 4820
4823 IF OBJX=15 AND OBJY=18 THEN GOTO 4820
4824 IF OBJX=15 AND OBJY=22 THEN GOTO 4820
4825 IF OBJX=26 AND OBJY=18 THEN GOTO 4820
4830 RETURN
5000 REM PAIXTIS
5010 ON PISTA GOSUB 5100,5200,5300,5400,5500,5600,5700
5020 RETURN
5100 IF X>15 AND X<26 AND (Y=10 OR Y=17) THEN X=OX:Y=OY
5199 RETURN
5200 IF (X=10 OR X=30) AND Y>6 AND Y<17 THEN X=OX:Y=OY
5201 RETURN
5300 IF X>12 AND X<28 AND (Y=6 OR Y=12 OR Y=18) THEN X=OX:Y=OY
5301 RETURN
5400 IF (X=10 OR X=20 OR X=30) AND Y>6 AND Y<19 THEN X=OX:Y=OY
5401 RETURN
5500 IF (X=10 OR X=30) AND Y>5 AND Y<18  THEN X=OX:Y=OY
5501 IF X>9 AND X<31 AND Y=6 THEN X=OX:Y=OY
5502 RETURN
5600 IF X>12 AND X<28 AND (Y=6 OR Y=12 OR Y=18) THEN X=OX:Y=OY
5601 IF (X=10 OR X=20 OR X=30) AND Y>6 AND Y<19 THEN X=OX:Y=OY
5605 RETURN
5700 IF X<12 OR X>29 OR Y<15 THEN X=OX:Y=OY
5701 IF X=15 AND Y=18 THEN X=OX:Y=OY
5702 IF X=15 AND Y=22 THEN X=OX:Y=OY
5703 IF X=26 AND Y=18 THEN X=OX:Y=OY
```

```
5704 IF X=26 AND Y=22 THEN X=OX:Y=OY
5720 RETURN
6000 REM ANTIPALOS 1
6010 ON PISTA GOSUB 6100,6200,6300,6400,6500,6600,6700
6020 RETURN
6100 IF XZ1>15 AND XZ1<26 AND (YZ1=10 OR YZ1=17) THEN XZ1=OXZ1:YZ1=OYZ1
6199 RETURN
6200 IF (XZ1=10 OR XZ1=30) AND YZ1>6 AND YZ1<17 THEN XZ1=OXZ1:YZ1=OYZ1
6201 RETURN
6300 IF XZ1>12 AND XZ1<28 AND (YZ1=6 OR YZ1=12 OR YZ1=18) THEN XZ1=OXZ1:YZ1=OYZ1
6301 RETURN
6400 IF (XZ1=10 OR XZ1=20 OR XZ1=30) AND YZ1>6 AND YZ1<19 THEN XZ1=OXZ1:YZ1=OYZ1
6401 RETURN
6500 IF (XZ1=10 OR XZ1=30) AND YZ1>5 AND YZ1<18  THEN XZ1=OXZ1:YZ1=OYZ1
6501 IF XZ1>9 AND XZ1<31 AND YZ1=6 THEN XZ1=OXZ1:YZ1=OYZ1
6502 RETURN
6600 IF XZ1>12 AND XZ1<28 AND (YZ1=6 OR YZ1=12 OR YZ1=18) THEN XZ1=OXZ1:YZ1=OYZ1
6601 IF (XZ1=10 OR XZ1=20 OR XZ1=30) AND YZ1>6 AND YZ1<19 THEN XZ1=OXZ1:YZ1=OYZ1
6605 RETURN
6700 IF XZ1<12 OR XZ1>29 OR YZ1<15 THEN XZ1=OXZ1:YZ1=OYZ1
6701 IF XZ1=15 AND YZ1=18 THEN XZ1=OXZ1:YZ1=OYZ1
6702 IF XZ1=15 AND YZ1=22 THEN XZ1=OXZ1:YZ1=OYZ1
6703 IF XZ1=26 AND YZ1=18 THEN XZ1=OXZ1:YZ1=OYZ1
6704 IF XZ1=26 AND YZ1=22 THEN XZ1=OXZ1:YZ1=OYZ1
6720 RETURN
7000 REM ANTIPALOS 2
7010 ON PISTA GOSUB 7100,7200,7300,7400,7500,7600,7700
7020 RETURN
7100 IF XZ2>15 AND XZ2<26 AND (YZ2=10 OR YZ2=17) THEN XZ2=OXZ2:YZ2=OYZ2
7199 RETURN
7200 IF (XZ2=10 OR XZ2=30) AND YZ2>6 AND YZ2<17 THEN XZ2=OXZ2:YZ2=OYZ2
7201 RETURN
7300 IF XZ2>12 AND XZ2<28 AND (YZ2=6 OR YZ2=12 OR YZ2=18) THEN XZ2=OXZ2:YZ2=OYZ2
7301 RETURN
7400 IF (XZ2=10 OR XZ2=20 OR XZ2=30) AND YZ2>6 AND YZ2<19 THEN XZ2=OXZ2:YZ2=OYZ2
7401 RETURN
7500 IF (XZ2=10 OR XZ2=30) AND YZ2>5 AND YZ2<18  THEN XZ2=OXZ2:YZ2=OYZ2
7501 IF XZ2>9 AND XZ2<31 AND YZ2=6 THEN XZ2=OXZ2:YZ2=OYZ2
7502 RETURN
7600 IF XZ2>12 AND XZ2<28 AND (YZ2=6 OR YZ2=12 OR YZ2=18) THEN XZ2=OXZ2:YZ2=OYZ2
7601 IF (XZ2=10 OR XZ2=20 OR XZ2=30) AND YZ2>6 AND YZ2<19 THEN XZ2=OXZ2:YZ2=OYZ2
7605 RETURN
7700 IF XZ2<12 OR XZ2>29 OR YZ2<15 THEN XZ2=OXZ2:YZ2=OYZ2
7701 IF XZ2=15 AND YZ2=18 THEN XZ2=OXZ2:YZ2=OYZ2
7702 IF XZ2=15 AND YZ2=22 THEN XZ2=OXZ2:YZ2=OYZ2
7703 IF XZ2=26 AND YZ2=18 THEN XZ2=OXZ2:YZ2=OYZ2
7704 IF XZ2=26 AND YZ2=22 THEN XZ2=OXZ2:YZ2=OYZ2
7720 RETURN
8000 CLS:MODE 2:PEN 1:LOCATE 3,19:PRINT"YOU SAVED YOURSELF BUT YOU DID NOT COLLECT THE
HEARTS": LOCATE 3,21:PRINT"PRESS SPACE TO TRY AGAIN FROM THE PREVIOUS LEVEL"
8005 LOCATE 3,23:PRINT"+1 EXTRA LIFE!"
8007 SOUND 1,60,8,14:SOUND 1,95,9,14:SOUND 1,119,8,14
8010 F$=INKEY$
8020 IF F$=" " THEN GOTO 8040
8030 GOTO 8010
8040 DIFF=9:score=0:LIVES=LIVES+1:GOTO 150
8100 LIVES=10:KRYFO=1:MODE 1:LOCATE 9,15:PRINT"YOU UNLOCKED 10 LIVES!!":LOCATE
9,17:PRINT"PRESS SPACE TO CONTINUE"
8105 SOUND 1,119,8,14:SOUND 1,95,8,14:SOUND 1,60,8,14
8110 F$=INKEY$
8115 IF F$=" " THEN MODE 1:RETURN
8120 GOTO 8110
8200 SOUND 1,119,10,14:SOUND 1,106,10,14:SOUND 1,95,10,14:SOUND 1,89,10,14:SOUND
1,80,10,14:SOUND 1,71,10,14:SOUND 1,67,10,14:SOUND 1,63,10,14:SOUND 1,60,20,14:RETURN
8300 SOUND 1,60,10,15:SOUND 1,63,10,15:SOUND 1,71,10,15:SOUND 1,80,10,15:SOUND
1,89,10,15:SOUND 1,95,10,15:SOUND 1,106,10,15:SOUND 1,119,10,15
8310 SOUND 1,179,10,15:SOUND 1,190,10,15:SOUND 1,213,10,15:SOUND 1,239,10,15:SOUND
1,253,10,15:SOUND 1,284,10,15:SOUND 1,319,10,15:SOUND 1,358,10,15
8320 SOUND 1,478,10,15:SOUND 1,506,10,15:SOUND 1,568,10,15:SOUND 1,638,10,15:SOUND
1,716,10,15:SOUND 1,758,10,15:SOUND 1,851,10,15:SOUND 1,956,50,15
8330 RETURN
```

YAMI NO RYŪŌ HADESU NO MONSHŌ

Year: 1986
Developer: CASIO

Platform: MSX1
Genre: RPG

Was it ever possible that after Knightmare's release and success, no one wanted to attempt the commercial coup with a clone that was truly up to it? Obviously not!

But there was a big problem. Knightmare for the time was a masterpiece; incredible soundtrack, fluid graphics, awesome gameplay elements designed with such mastery as to seem real, incredible use of depth etc... Anyone who had challenged him could hardly have avoided a great figure, so much so that a real playful colossus came forward. Someone who at the time had the economic strength to wipe out anyone, both in terms of hardware and software... CASIO!

You must be wondering: did Casio succeed? The intentions were good, but trying to fight a game with a clone is already half of a failure.

The game was released in Japan under the title Yami no ryūō hadesu no monshō, renamed Leonidas in some countries, Crest of the Dragon King Hades of Darkness in England, The seal of Hades in the United States and Crest of the Royal Family in Korea... In short, it had a name for every occasion; at least one thing was certain... the project was destined only for the MSX market. This purpose we understood very well.

Despite a very articulated and well-made soundtrack, fluid scroling and decent graphics, unfortunately the game did not sell as expected; too similar to Knightmare and too difficult. Worst of all, it was an unprecedented commercial mistake... Knightmare (the original one) came out as a distribution for sale between December 1985 and January 1986, just in time for Christmas gifts, while Casio's announcement of the game and its future sale came as well.

The guys who bought Knightmare totally lost interest in playing one of its clones. Leonidas, which was made in a hurry to get out in the summer of 1986, also had negative ads from the major magazines that shouted at plagiarism.
The game was forgotten in a moment and it fell in oblivion. This explains why we find very little information about this game and many of you hear it mentioned for the first time. Although, without Knightmare in the way, it would have been a timeless bestseller and acclaimed by all the players on the planet.

Now let's move on to the curiosities; Casio presented it as the sequel to "The Stone of Widsom" (its 25th game) which had been released simultaneously with Knightmare in early 1986. But The Stone of Widsom was a completely different game; it was an RPG, very nice, full of dungeons and with lots of stages. A great game, but it had nothing to do with either its pseudo sequel Leonidas or Knightmare...

Leonidas has six levels, slightly longer than the Konami game and very difficult to complete. The game is frantic and very fast but calibrated so that sometimes it seems like you can win easily and then die miserably. It forces you to go get the calendar to remind you of the name of all the Saints... :-)

If you liked Knightmare, it's a great diversion for real games.

You can find it and play at the following site:
https://www.file-hunter.com/MSX/index.php?id=leonidas

by **Ermanno Betori**

OUR FINAL SCORE

» Gameplay 90%
High playability. Graphic design makes good use of the MSX1 computer features. Sound with a well-kept musical motif that accompanies the phases of the game well. There are various bonuses to take among which the reserve of oxygen in fact our hero looks like a diver or similar.

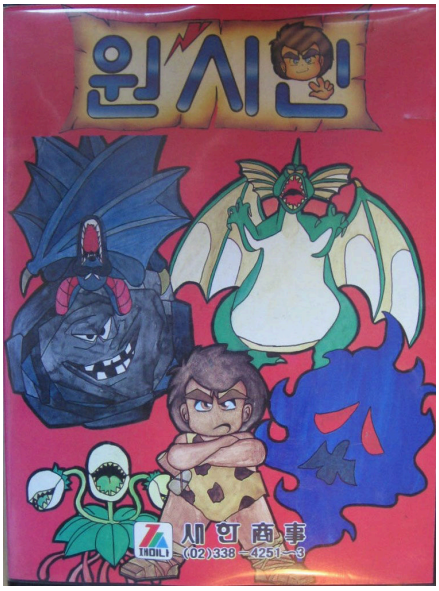
» Longevity 70%
A difficult game; an arcade novice can't even finish the first level. But it's not impossible, even if it's really tough in some places.

WON-SI-IN

Year: 1991
Developer: Zemina

Platform: MSX1
Genre: RPG

Another Knighmare clone, which we are now going to examine, is a very rare game produced by Zemina, a South Korean company. In 1991they released a game called WON-SI-IN which roughly translates with “prehistoric man”.



Won Si In is a porting of a Famicom game named “Shin Jinrui: The New Type” (published as Adventure of Dino Riki in the US), which apparently gave Zemina the opportunity to exploit their vertical scrolling code in porting the title. They did it so well, that they even recreated the instability of the action which is quite visible in the Famicom original.

Despite some other small bugs, the porting is very good. Even the Dinosaurs figures were not changed by a single pixel and the enterprise was not at all simple. However, Won Si In ends after just three full levels, leaving out the final challenges and the final clash with the supreme boss, who actually looks like Mario Bros' Bowser!

The game features a caveman to survive along a path steeped in difficulty, upgrading his spear weapon to stone axes, boomerangs and even flamethrowers. Though not an easy game, Won Si In is never as difficult as The Three Dragon Story.

In early 1992, Zemina had announced the sequel Won Si In 2, which was not completed before the end of the alliance between Saehan Sangsa and Zemina. Kim Eulsuk left and with his staff formed the Open company. They made the sequel on Master Systems such as Wonder Kid/ Adventure Kid which in fact appears to be a clone of Wonder Boy.

For retro gamers, collectors and enthusiasts, the cartridge today has a very high commercial value and moreover it is truly untraceable, which has contributed to the oblivion of this game.

Careful graphics that take full advantage of the features of the MSX1 computer. Sound: repetitive but well-kept musical theme, perhaps not properly suited to the game's storyboard... Yes, the following levels may feature new tunes, but I repeatedly get annihilated in the first stage!

by **Ermanno Betori**

OUR FINAL SCORE

» Gameplay 88%

A really hard game, but Knightmare seems like a piece of cake in comparison. Recommended for real Super Arcade Men!

» Longevity 70%

Three levels only, but the difficulty will make them last a long time... Unless you get frustrated much sooner!

CYBORG-Z

Year: 1991
Developer: Zemina

Platform: MSX1
Genre: RPG

Let's conclude the series of Knightmare clones reviews by talking about the plagiarist software house “par excellence”: Zemina!

One of their rarest games of all is called Cyborg-Z. Also released in 1991 on MSX, it was one of the last games developed on the 8-bit platform by Zemina that did not reach much public interest. The majority of customers were already mistreating their joysticks on more powerful machines at the time, and the circulation of this game was quite limited.

The only known footage proving the existence of this game, previously considered lost forever, came from a Japanese owner. A group of enthusiasts then began a sort of treasure hunt that ended with the discovery of the original cartridge. The software house had finally switched to the Megabir ROM (equivalent to 128 KB), so this game includes much more graphics than the previous Zemina products.
Cyborg-Z is a vertical scrolling shoot’em-up, very similar to Knightmare. Even the gameplay is incredibly similar with only a few differences.

The main character is a copy of the giant Mazinger Z from the 1972 anime series, including its Rocket Punch attack. The game has the same weaponry system seen in The Three Dragon Story, with multiple weapons that can be switched, but it's much easier here because the mecha is given many extra lives. He also has got a power bar and a speed bar; the power bar runs out when the character is damaged by enemies. Most weapons, except the rocket and the laser beam, have limited ammunition. There are two types of invincibility elements. The red invincible element "α" can kill enemies, but the blue invincible element "β" cannot. Each phase features a central boss and a final boss to deal with.

After passing each stage, we will find a bonus roulette wheel. If the arrow lands on a yellow space, you can select an item and you can repeat it endlessly by always ending up in yellow.

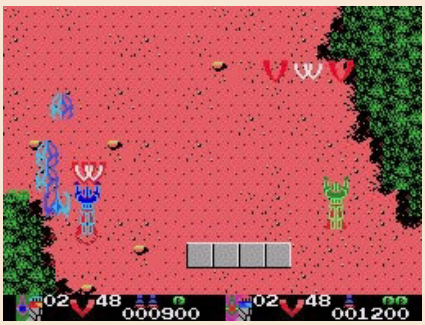
There is also a dual-play mode for two players. What's special about this type of game is that some of the background music in the game is composed of classical music.
The sound engine of this game was “stolen” by Hi no Tori Hououhen: Gaou no Bouke.

The game has a hidden command which works as a "Sound Test". You can access it by pressing the sequence Up, Up, Down, Down, Left, Right, Left, Right (most of the "Konami Code") on the Play Select screen.
A version for SG-1000 was also released, basically a direct porting from MSX

The game was programmed by: Koo Eun Joong (scenario) also curated the scenes of Woonsin, the other Knightmare clone, Ji Eun Kyung (graphics), Kim Sung Keun (music), Koo Eun Joong (programming), who also was Brother Adventure's father, Kim Eul Suk (direction) who also directed Super Boy 1 and 2.

This team of programmers in Zemina was called “Magic Tiger”.

by **Ermanno Betori**



OUR FINAL SCORE

» Gameplay 93%

High playability, unsurprisingly the same as in Knightmare!

» Longevity 90%

Difficulty well balanced through the stages. You’ll play it again and again!

ELLENICA

by Carlo N. Del Mar Pirazzini

The 80s/90s console scene is really very active.
Over the months I have witnessed several homebrews, physical games and projects under development for numerous consoles.
Some time ago on RMW we also talked about NESMaker and how this is helping developers create their own games even with little programming skills.
I recently came across an all-Italian project, known through a Facebook page that immediately intrigued me.

I'm talking about **Ellenica - Dusk of God**, Ultima style game in "creation" for MEGADRIVE.
I was amazed at the development of the game and immediately contacted Ellenica's "dad", Sefano Canali.
After a brief presentation by post, we agreed to make this product visible to our Italian and English readers and to follow it throughout its creation, supporting it.

But I give the floor to Stefano in art McValdemar who will tell us about the development of the game.

NTH - Tell us a bit about yourself and how you approached this world. What you do in life and your passion where it comes from.
MC Valdemar - Hi all of you, I am 47 years old, I work as a sales manager at a company that deals with Cloud services for European agencies.
Video game from the beginning (started with Videopac G7000, touched all 8 bits Commodore and moved to the PC. Consoles have tried several offers: MD, SNES, PSX, DC, XBOX/360, PS2/3/4, GC.
Let's just say that the reference console for me remains the MD for a matter of how I lived that time.
You dreamed in front of the cabins, you dreamed in front of the many cartridges available in the stores and you could only buy 3 or 4 games a year and you ended up knowing them by heart.

NTH - Do you have any other projects in mind besides the beautiful Ellenica?
MC Valdemar - I have always touched on development but of relevance I can only recall:
- PC 8086 games (CGA/Hercules) published on PC Games (a magazine of the time)
- followed by the text version of Infocom's first graphic adventure: Return to Zork. I wrote it with permission from Activision. It is published on IFDB.
- last year I picked up the C64 and I baked a SEUCK game with extra additions and won the SEUCK 2020 competition: Pagoda Warriors 2.
- In the last 5 years I have collected material, ideas, worked on tiles, prototypes for my idea of Rpg/adventure mix that is Ellenica. The development only recently I decided to move it to MD but, with the help of the creator of SGDK, it seems that so far things are going well despite the challenge in terms of memory.
Obviously, like everyone else, I have an HD full of dead prototypes of games designed, sketched and never finished.

NTH – I think it's time to talk about Ellenica. Freewheel.
MC Valdemar - Ellenica was born from a project of mine, designed for PC, several years ago on which I worked directly, and paid freelancers, but on which I never reached a tangible milestone because all the work was done mainly in writing history, thinking about the world to offer, working on tilesets, etc... and little on code.
In February 2020 the idea of trying to compress an RPG conceptually designed for PC into a 32Mbit cartridge for MegaDrive, thanks to the existence of Stephane Dallongeville's exceptional SGDK that greatly simplifies video and audio management on the 16-bit SAW.

THE STORY
The game is set in Crete in the year before the outbreak of Thera (now Santorini), an eruption that will lead, together with the invasion of the Mycenaeans, to the disappearance of the Minoan era and the closure with the Mycenaeans of the Bronze Age of Greek history and the beginning of the Dark Ages.
The inevitable event, the link with the "true" story, the

proximity of the concepts of the eras with those used by Ultima, a title that is given by a Latin word in my opinion was the right mix.
This year the player is the victim of one of the earthquakes that preceded the explosion. Surviving the earthquake, he will unfortunately discover the loss of a loved one and from there begins the adventure that will see him chasing the faint possibility shown by a dream, to bring the lost person back to life.
The events he will encounter on his way will, however, allow him to make actions and choices that will bring him closer to the gods of the Parthenon who seem to promise him this possibility, or the belief towards the Goddess Mother to whom the lost person belonged.
In a duality that, in intent, a little reminds us of the evolution of the events of Avalon's The Mists.
There are clashes, there are 4 mythological giants (of which only two are mandatory) but the emphasis is more on exploration and investigation than combat.

THE GAMEPLAY

Personally I'd like to call Ellenica an "ARPA", an acronym for an Action Role Playing Adventure.
This is because I believe that RPGs are a very interesting kind of game, not having, more than other genres, a "unique" recipe. They are more like a cocktail where even just changing the balance between history, characters, battle systems, puzzles, freedom of exploration generates seemingly similar but, played, substantially different titles.
Ellenica starts from being similar to a tile-based 2D RPG, where exploration sees a general map (overworld) and overworld-accessible locations that operate from detailed leases on a different scale. The concept is very similar to the first Ultima (pre Ultima 6) and we have that the island of Crete is the overworld and detailed rentals, ranging from cities to caves, from cemeteries to abandoned ships, from dungeons to clearings with particular monuments, are partly available at the beginning and partly unlocked with history, or following a dialogue or reading of a writing. Some are "required", others are optional.
Each map, both general and detailed, consists of one or more screens, this was necessary in order to have very different tilesets and be able to make everything work in the limitations of the console.

The exploration takes place in real time and the interaction is possible both with some elements of the seabed, such as doors or trunks, and with objects that can be collected and used or consumed.
By approaching an enemy, either by enemy aggression or by our choice, we can start a fight that transforms the game more like the old SSI titles or a round of Advanced Hero Quest.
A grid appears and the fight is in turns. Start the player by moving a number of boxes according to the speed STATE, then use ACTION POINTS to attack, change weapon or consume an object, decide whether to use it for a defensive posture and then pass the turn to the various enemies.
Each attack clearly considers the outcome of a dice throw (including critical and fumble shots), the skill with the class of weapon used, and the positioning of the two fighters with bonuses for side or shoulder attacks. Damage is measured from cut, puncture, blunt, throw, and "magic" damage and compares with all of these categories of enemy defense to see how it passes.
There is no magic in the game except in the form of weapons of the gods (usable) and monsters that have magical abilities.
STAT and SKILL management, object usage, quest and map work as in Skyrim. START opens a menu with four main entries from which YOU CONTROL STAT/SKILL (in addition to experience levels, player status, hunger/thirst), items to wear armor/weapons, consume items such as food/beverages/potions, items that can be read, keys and miscellaneous items (often this item or item useful only for trade).
Leveling earns points to distribute on SKILLS improving attack, defense, robustness, luck (in dice), etc...

In short, a game quite different from the more than 1000 titles available for MegaDrive because "very PC".

So far the main challenges are:
1) It's my first MegaDrive title
2) 32Mbit (4MB) are very few considering that the title has hundreds of game/animation tiles. And each game tile corresponds to 5x5 tiles of the MD (for the bottom) and 8x8 tiles for sprite (standard).
3) The work proceeds in the free time I have, which is

difficult to predict and has ups and downs. Realistically, however, I should have a DEMO ready by the end of the year where I will try to have most of the gameplay ready, so that I leave 2021 only for balance/content.
The DEMO will roughly contain only 20% as content but 85% in terms of gameplay possibilities because those are the things I want to test.

I would also like to mention the team of freelancers whose professionalism I have been using for a long time:
- Nathan Skaggs develops graphics on PC, graphics that I then take to MD to adapt resolution, colors, etc... according to THE limits of THE VDP. The more I take care of all the additional graphics for items, interface, digitizations, ...
- Al Riad for hand drawings that are then digitized, compressed into 320x184 images because particular locations have images and additional text as if it were an old adventure/Book Game.
- Walter Torbito who took the story and enriched it with situations that will put the player at the “intimate” crossroads of following the gods of the father-in-law or remaining tied to the goddess mother
- Trevor Lentz for PC music compositions. That may never be made public but, MB permitting, they could pop up in part as digitizations.
- Paulo H. G. Ferreira, who is editing the music of the MegaDrive version.

Currently there would also be a publisher who showed up for the physical version but this is something I prefer to evaluate only when I have the consolidated DEMO at the end of the year. Both because I can't make a commitment (being a "hobby") but also because the investment so far has been all mine, especially with freelancers, and although it was clear right away that I could never recover with a MegaDrive title, I don't even want to throw away what I did.

My idea, before discussing with this publisher, was to make THE ROMA available on Itch.io and side by side a limited 25/30 copies as an “old-fashioned” physical release, i.e. not only case, cartridge and manual, but also map of cotton, beast, player's card and coin metal like the old Rpg.

Unfortunately, the MegaDrive scene is pleasantly alive but also linked above all to “bite and run” experiences, some more arcade titles, puzzle games, where in a couple of hours you can see much of what the game can offer and this is another obstacle to a game like Ellenica. But by doing it out of passion and not for gain, I had no choice.

A PC version that offers the entire game as it was intended could resume to be developed following the MegaDrive demo. If the feedback shows interest, I will entrust development to a couple of people I know for a Unity version.

We thank Stefano Canali for his availability and invite you to follow us for further developments and a “road” test.

Also support and follow the channels where you talk about the game to the following addresses.

Where to follow the project:
1) Twitter: https://twitter.com/McValdemar
2) Website: http://www.segamegadrive.it/ellenica/
3) Itch.io: https://mcvaldemar.itch.io/

MC Valdemar - Thank you and see you soon!

Personal Opinion
Unfortunately I was not able to test the game by hand, but from the photos present and the video viewed on the reference pages I can say that my personal EXPECTATION is really very high.

Especially from the video we can see the attention to detail and fluidity of the product even at this early stage.

Congratulations to Stefano and his group for the passion and love they employ in the development of a product that like RMW we will follow very carefully in the coming months.

ELLENICA - last minute's call...

We were about to close the number when we received an email from Stefano Canali with a series of additional screenshots of his game.

All the images were also accompanied by captions that complemented the information we had given in the article dedicated to Ellenica.

Obviously, we could not avoid to publish all this material, which, if possible, will help us and our readers to get their mouths watering.:-)

Figure 1 - the opening menu of the game with options that will allow you to start a new game, continue from the last save game, see the controls (pad 3 or 6 keys) and a summary of the story.

Figure 3 and 4 - These are two overworld exploration screens. The exploration takes place "in rooms" but the rooms are often wider than the screen so they are cut off in 4 directions. In some cases there are icons that show interesting locations in the room that can be: cities, single houses, dungeons, caves, ruins, wells, wrecks, etc... that are explorable. The sea as well as the clouds are completely animated.

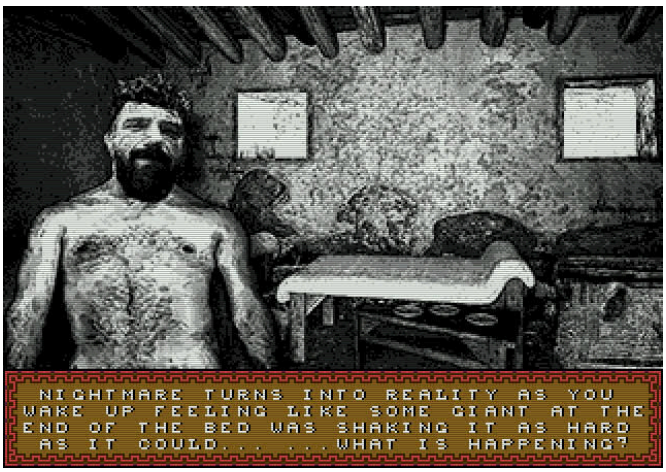


Figure 2 - the game has several B/W images accompanying some moments of history or particular revealed locations.
This is one from the first moments of play.

Figure 5 - Except when in combat you can access a menu that, compared to my preview that I sent you, has been enriched with a tab.
In this case we see the MAP (map of the main discovered locations).

Figure 6 - The other areas of the menu are: STATS (see stats, skills, status, money, experience points), ABILITIES (see hero points that you can spend on active talents, consuming action points, or passive), ITEMS (see, exams, uses, wear items, read books, etc.), QUESTS (keeps track of these and their steps).

Figure 7.8 - A very large part of the elements of the playground come from drawn versions of real elements of the Minoan or ancient Greek era. In this case, the portal that functions similarly to Ultima's portals, exists in Naxos and is called Porta del Sole.

Figure 9 - Combat is in turn mode, similar to an Advanced Hero Quest round. From the game options you can decide the level of information of the combat menus so everyone has their own level of information; more invasive or minimal.

Figure 10 - And finally a screen from a first boss (in a test area). For those who remember the Game Book it will not be difficult to see the tribute to the Sortilegio saga.

Figure 11 - A prototype of the cartridge, where you can see the graphics that will also be on the cover of the game, and the main inspirations: The SSI games, history books of Minoan culture, Ray Harryhausen (and films such as Argonauts, Titans' Encounter, etc...) and the adventure part/drawings of the Game Books.

STURMTRUPPEN

Publisher: Leader
Developer: Idea
Year: 1992
Genre: Platform/Shoot'em up
Platform: Amiga

Achtung! Achtung!

In 1992 Idea Software developed a game based on the stripes of the German soldiers of Bonvi (a great Italian author unfortunately missing) for our Amiga.

The Idea software actually had a real passion to develop under official license from comic strips. A few years before they published Lupo Alberto and at the same time they was about to release Cattivik.

The game, based on the laziest soldiers in the comic book universe, came out at the beginning of the summer season collecting not always positive opinions in the magazines of the sector.

But let's take it one piece at a time and start by explaining what we're going to face.
The player controls a plain soldier of the Sturmtroops, on foot or driving vehicles against the troops and vehicles of the Sturmtroops themselves (which is quite absurd not to have real enemies, ed).

Everything moves horizontally from the right and the goal is the most classic, reaching the end of the path. The development of the game is on six levels with different settings. You go from the city at war, to the ruins, to the barracks, to the desert and even to the alpine and glacial landscapes.

Let's start as we said with a soldier resting in the most classic style of platform games. Initially we can only kick and punch our opponents. From some eliminated opponents we will be able to collect weapons of various types (gun, rifle and machine gun) but with unlimited ammunition.

When we take over a vehicle along the way, the view moves away and the game becomes the classic shooter with unlimited ammunition.

When we recover an off-road car or motorcycle, the game reminds us of the classic Moon Patrol. At each different game stage, opponents will also change and the challenge will be balanced to the level we will face.

Our soldier has only one life, but he can take several blows. The energy bar is represented by a shaped shape of the soldier himself that turns, every time we are hit, into a skeleton. Vehicles have power, too, and if they're destroyed, we'll be back on our feet.

In short, to look at it this way the game seems tantalizing and reminds the distance of an Italian Metal Slug.
The graphic sector is very pleasant and well made over time, although Amiga was never fully exploited; in the final view the product is not bad even now.

Less enjoyable is the audio department with almost non-existent sound effects and repetitive music that will be annoying during the game.

So can we compare it to Metal Slug? Nein! It lets himself be played and the game is nice thanks to the license of the comic book of the late Bonvi, but it is rather short and in a short time you will find yourself at the final screen.

The Amiga package of the game featured and still features two floppies and a nice manual in Italian with five incomplete Bonvi comic strips. Why are they incomplete? Well, the purpose of the game was to get you to finish the levels so you could see the ending of each mini comic book.

To sum up, we can say that it is a niche product for the fans of Italian comics, perhaps made too quickly to be released on the market.

If you are a fan of Sturmtruppen you will enjoy to play with it from time to time, everyone else will soon leave it on the shelf (or in the directory of some emulator).

by **Carlo N. Del Mar Pirazzini**

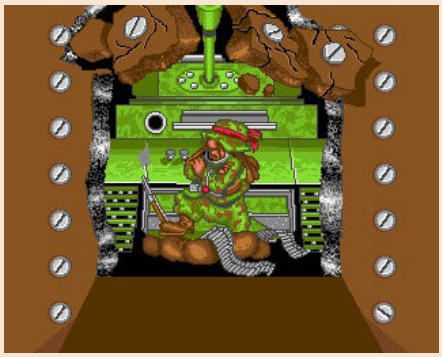
OUR FINAL SCORE

» Gameplay 75%

It is certainly easy to be playe and I would say the gameplay is also well developed as is the graphics. But it lacks something.

» Longevity 60%

The lack of "something" that I told you before, makes this product too short and easy to be finished without too much effort. If you're a fan of Bonvi and the Sturmtroopers, maybe you'll play it again once in a while, otherwise it'll fall into terrible video game oblivion.



WONDER BOY DRAGON'S TRAP

Developer: Lizardcube
Publisher: Dotemu, Nicalis, Headup Games (with Sega's permission)
Genre: Scrolling platform game
Platforms: PC, Nintendo Switch, PS4, XBOX, IOS, Android
Platforms since1989: Sega Master System, Game Gear and PC Engine

"Even after many years, pearls shine like freshly harvested."
In 1989 Sega released the sequel to the playable and wonderful Wonder Boy in Monster Land, an arcade game later converted to any home system. It was released Wonder Boy III the Dragon Trap, but only on the Sega Master System (and only later on PC ENGINE). It was such an incredible success that it was recognized as one of the best titles on the small 8-BIT SAW console.

In 2018 Lizardcube dives into an almost unlikely new adventure: creating a remake of the original game but doing so in the spirit of modern times. With the support of Ryuichi Nishizawa to ensure that the original work is respected, the Lizardcube group today allows us to play, or rather replay, an already wonderful chapter of video game history.
Let's find out together if this new "experiment" went the right way.

As we said, this is not the first episode of the Wonder Boy saga, but the third (it would be better to say the fourth, but I would like to forget the terrible Wonderboy in Monster Lair). Everything is filmed at the end of the second episode (the aforementioned Wonderboy in Monster Land) with our little hero Tom-Tom preparing to fight the terrible Meka Dragon. Once defeated, the monster before he dies will cast a curse that will turn him into a small fire-breathing lizard with the obligation to escape from the ruined castle and seek throughout Monster Land the cure to become human.

The journey unfolds in a world far from linear. We will not have levels to reach left or right, so there will not even be subdivisions into stages or levels. Depending on the course of history and the special abilities of transformations and exploration, we will have to discover the way forward.

The game presents itself as other classics of its time (Zelda II, Simon's Quest, Metroid) and today could be identified as a classic Metroidvania. The novelty that struck at the time was the transformations of the protagonist. Every time we defeat an end-of-area boss, he curses us and turns us into a different creature with different characteristics. And so from a fire-breathing lizard we'll be turned into a mouse, a piranha, a lion and a hawk. Each of them will present a different gameplay and unique features that will open the way for us to continue in the game.

For example, the mouse will allow us to climb and walk on the walls and sneak into unthinkable places, the Piranha will allow us to swim in the water levels, the Lion with its strength will allow us to break down otherwise unshakable walls and finally the hawk will allow us to reach some areas of Monster Land impossible if we do not fly.

As we said, the non-linearity of the game allows us to explore almost everything freely, but if we are poorly equipped or not suitable for that area, our existence will be reduced to the minimum.
In this case, in addition to the various transformations, you will need to equip yourself in the many visible and secret stores in the game where you can buy weapons, armor and shields, healing potions and more.

But we arrive talking about the realization of the game. The question is twofold.
First, is it a time-resistant game? Thirty years have passed since the first version, can it stand the comparison with today? The answer is DEFINITELY YES! The captivating gameplay, the exploration of the world and the fresh way of managing the game action still make DRAGON'S TRAP a small pearl.
The second question instead speaks directly to the developers. Did they do their job right?
Affirmative! They made the game really superlative from a technical point of view. The high-definition graphics are wonderful, redesigned with a fairytale and cartoon style that delights the eyes. Not to mention the animations of every single element of the game. All the transformations of the protagonist are exceptional (the falcon's walk/jump is beautiful) and everything moves perfectly whether you look on your PC or you do it on Switch.
The technical implementation remains at very high levels, even on the music that immerses perfectly you in the game and does not disfigure in any way with the originals.

And precisely speaking of the originals, we must mention the possibility of switching during the game from the modern version to the old version of the Master System, including music. All by clicking a button on our joypad. Great effect for all of us nostalgic.
The level of challenge always remains balanced and never too punitive, but it requires an effort of attention on how to deal with the challenges and the final bosses.

Each final monster presents with well-defined features and can always be tackled with a perfect use of correct equipment and precise shots.

Creating a remake of a work has rules and the main one is to preserve the original spirit and this has been done impeccably.

The first impact is staggering. A work that speaks not only of the graphic or sound sector, but also of the character's characterisation. A protagonist who has undergone excellent treatment in his transformations and in the aforementioned animations.

In conclusion, Wonder Boy Dragon's trap is an excellent product. A product that does not feel even a second of the past 30 years thanks to aesthetic improvement and does not betray the spirit by one gram.

Three levels of difficulty, the chance to play the adventure like Wonder Girl, the addition of a fresh gameplay, honors the Lizardcube boys who demonstrated love for the original work, bringing it back to our screens to make us children,

but also to act as a link between us dreamers of the past and the new generations.

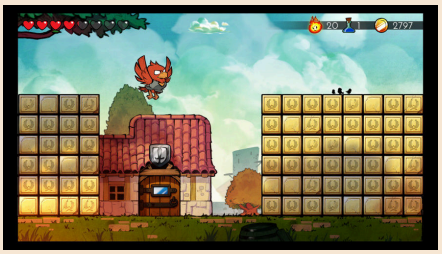
...I'm gonna go. I have to go back to Monster world to dream again... And again... And again... Have a good trip, Wonder boy.

by **Carlo N. Del Mar Pirazzini**

OUR FINAL SCORE

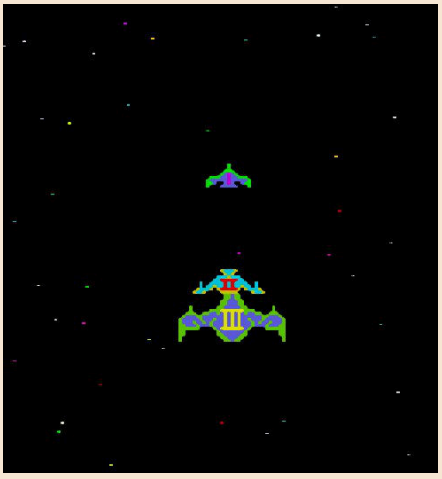
» Gameplay 95%
Superlative gameplay in 1989 refreshed by first-class technical support. Each character will provide hours of fun and exploration. Not to mention the selectable difficulty levels that raise the stakes and the possibility to face the adventure in a female way. Never punitive, severe but fair in facing challenges.

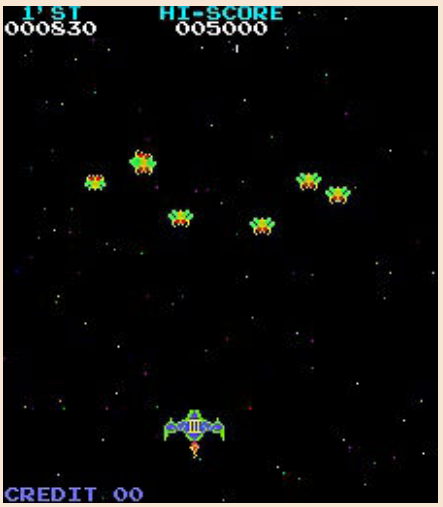
» Longevity 90%
Not very long to play but must be carefully explored, then you'll like it even after the first round.



MOON CRESTA

Year: 1980
Publication: Nichibutsu
Platform: Arcade, Commodore 64, ZX Spectrum, Amstrad, Wii, Playstation 4
Reviewed version : Arcade
Genre: Shoot'em up





We kids of the '70s had a lot of luck in my opinion. We lived through the era of the Japanese robotic invasion when a prince from another world inside the screen of our TV every day worked hard to defend our planet from the invaders of Vega. When an unknown and ancient civilization of the past rose from the Earth to regain a lost kingdom and when in our young teenage mind we dreamed of making cybernetic connections to become powerful robots.

A period, that between the late seventies and the mid-eighties where, thanks to phenomenal machines coming from America, they could realize our dreams in fantastic realities, always inside the TV screen but under the orders of a controller that we could take in our hands! In those years when Japan was able to regain its post-nuclear revenge by conquering the world peacefully thanks to the invention of gaming consoles and when America, after sending men into space, began to resell those fantastic technology ideas by sending them to our homes in the form of personal computers.

Even today, and you readers know it well, despite the fact that video game technology has almost reached a visual real experience, a game of the classics of the past has become a pleasant habit. So you can taste once again the entire video game story since few pixels were controlled on the screen to today's extraordinary three-dimensional open-world worlds. But the amazement associated with those late '70s booths is still alive inside our minds when every little step forward in gameplay was literally a big step for the video game media.

So after endless battles with the cabin of Space Invaders where our only defenses were those barriers that second after second crumbled under our blows and under the lasers of aliens, Moon Cresta gave us a modular spacecraft composed of three stages (with the number clearly imprinted on the shuttles themselves) as well as a new invasion of alien ships that, although arriving in a group, knew how to move in every direction using even simple and sometimes predictable patterns of movement with the sole purpose of reducing all our modules into small scrap scattered in space.

In Moon Cresta we started with shuttle number one, small size and possibility of shooting one shot at a time. A ship with little firepower but with the ability, given the size, to shell like an eel into the alien rays.

If you were hit, you would immediately move on to the control of ship two, my favorite, small more or less like the first but equipped with a double shot capable of making us believe that they had become immortal while destroying enemy ships one after the other. The third part of our ship, perhaps the most hated by players, was always equipped with double firing but also of a huge tonnage that unfortunately for us also became an easy target for rockets and enemy meteorites.

But where was Moon Cresta's innovation besides not being a static (albeit fabulous) shooter like the old Space Invaders ? Well, if you got past an unscathed alien horde, you could

hook up with the module immediately following the one used with the Jeeg Robot sequence! To do this, we had to match the docking points exactly so that we could join the two ships and get a new one with increased firepower!

Simple but brilliant considering that we were talking about the first time you could do such a thing on the screen of a colour booth, at the cost of 200 lire per game, while the others played cards and smoked cigarettes inside the bar where until recently you could only play pinball and billiards!

The maximum obviously was to be able to recompose the ship with all its components in the correct order thus obtaining a fire power out of parameter (the shots of the three ships were added).

Obviously the sense of omnipotence did not last long enough to realize that so the ship as well as offensively effective was still too big to resist for long the treacherous and increasingly lethal enemy attacks. In fact, the game always proposed the same levels in sequence that once passed they had to face again with a greater level of difficulty.

by **Flavio Soldani**

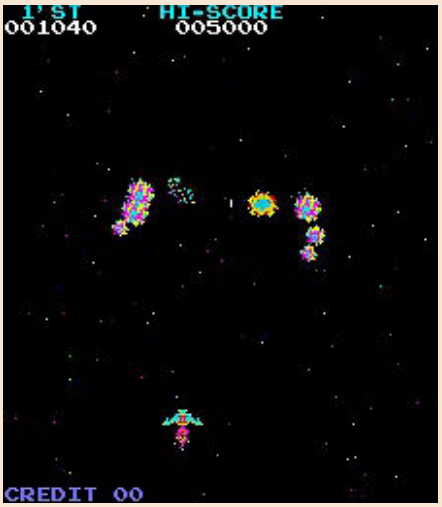
OUR FINAL SCORE

» Gameplay 89%

Frenetic and entertaining, we will fight starships, meteorites, rockets and alien formations of strange and menacing shapes. Collisions sometimes approximate especially at the highest difficulty levels in favor of CPU of course.

» Longevity 80%

Moon Cresta is always fun to play even if it belongs to the period of the big bang of space shoot'em-ups (and of the entire gaming industry). The games as in all the Arcades of that time are intense but fast but in my humble opinion it is still advisable today especially in its Arcade version emulated by MAME.



RAGING BEAST

Year: 1985
Developer: Firebird

Platform: Commodore 64
Genre: Simulation

Even on these new hot, post-pandemic days, my attention could not stand still and I was struck by one of the many singular and underrated games at the time, typical of Italian newsstand tapes. And that's where I found him, in one of the many I had, simply called “Olè”. The presentation of the game had the screen of a bullfighter intent on his work, that is, challenging the bull by waving the red drape. At the time, I played it for no more than five minutes, given the complexity of the controls and the practically non-existent gameplay (at least for me).

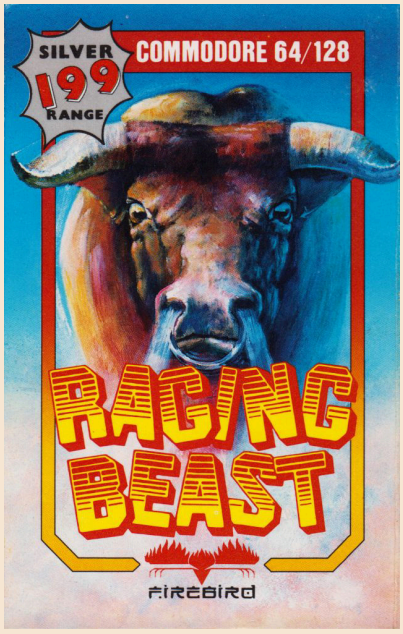
Over the past few months, I've discovered that his real name was Raging Beast, so I've decided to play it seriously and learn the game mechanics as well.
During the charging I wondered if the game would incite violence on animals, in this case on bulls in bullfights, but it did not seem so because in the game you do not notice the slightest violence except when you are mistaken and the bull crowns you. Then the stretchers arrive to transport the poor bullfighter to the hospital and a poster is published on the main square with the name given at the beginning of the game.

The game is presented with a fixed screen that represents the entire arena with the noisy audience. The bullfighter and the bull are a little small shape sprites, but very well drawn and with smooth movements. As I said, the difficulty lies in getting acquainted with the joystick commands and remembering the actions to be performed; for example, right lifts the drape, left shakes it to provoke the bull...
The aim of the game is to score as many points as possible in addition to winning the bullfight by not killing the bull, but by putting on a red crown thrown by the audience, to get them both out of the arena safely. In addition you can also ride the bull as in the rodeo style; very funny and no less easy than in the reality...
Being a one-level game, you might think that it can get tired within a few days and instead I assure you that once you learn it, you will take pleasure and have fun, not only alone, but also challenging friends and family!

I must admit that Commodore 64 had titles for all tastes and was able to represent anything that came to the developers' minds (... and it is still happening! Albeit at a slower pace of course). This was one of the peculiarity of the newsstand cassettes, in addition to finding cheap games in Italian.
At the beginning of the article I mentioned a detail that captured me a lot, namely inserting the name before starting the game... Someone might object, "What will it be? Many games did, especially sports ones. ", but here the name serves to create a poster and a newspaper's article's title in Spanish (language not very popular on Italian Commodore 64 games). Depending on our work it appeared as an article of praise or a provocative one.

For those of you who will be home in these two months of vacation, I recommend playing it, even with your friends.
For others, remember that when you return home, perhaps you will feel less the lack of beaches with plexigass!
See you in the next issue.

by **Daniele Brahimi**



OUR FINAL SCORE

» Gameplay 60%
Not easy to be mastered, it requires some time.

» Longevity 70%
Gives all its best played with your friends.

LUCI DELLA FINANZA

(The lights of finance)

Year: 2020
Developer: Marco Vallarino

Platform: Multi-platfom
Genre: Text Adventure

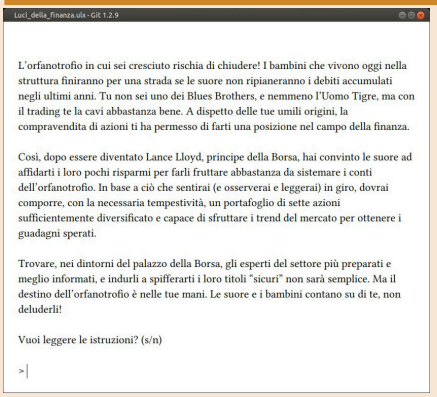
Language: Italian only

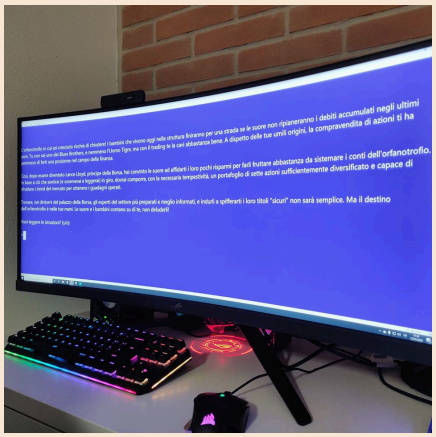
Summer is finally here and, despite the many precautions we adopt in this post-lockdown, it is time to have some fun. Have you already reached your favourite beach or sea town? Have you finished digging ditches and laying barbed wire to defend your 7 square meters of safety? And are you finally looking for some refreshment? Well, we have the game for you.

Taking advantage of the forced lockdown caused by COVID-19 (may God strike it dead!) and trying to transform into opportunities the restrictions we have been subjected to in the past months, our dear Marco Vallarino has decided to get back on his keyboard and bring out a new text adventure as a present to all of us, fans of this kind of entertainment.
The plot is very simple: as Lance Lloyd, a man of humble origins, now a successful trader, we must find a way to save the orphanage in which we grew up and thus help the sisters who lovingly raised us. Unlike the Blues Brothers, however, we do not have the slightest talent for music or theft, but we can use our exceptional business sense to play on the stock exchange and generate, starting with only 7 shares, enough profits to save the orphanage. After convincing the sisters to entrust us with their savings, we begin to travel in a world of play that is not at all wide but more than enough to offer a good degree of "immersion" in the plot, looking for clues that allow us to identify the most lucrative businesses and conclude them before it is too late. In fact, timing is one of the key elements to win and, as in real finance, a good deal has to be closed at the right time, under penalty of a decrease in the profitable amount.
The mechanics are typical of treasure hunts, featured in many text adventures since their inception (as seen in Colossal Cave, Zork or Adventureland) although here the game development is a bit atypical because, instead of looking for an object and taking it back to the place intended to receive it (the trophy board in Zork or the inventory in Adventureland), here the "treasures" are all in one place (the Stock Exchange). You just need to know how to recognize them.

The need to increase our selection capacity will be the cause of our journeys to the City: identifying the most lucrative stocks requires first-hand information, which can be found wandering around the City and interacting with the characters who populate it. Once we find out what is worth having in our portfolio, we just have to run to the Stock Exchange before others come to the same conclusion and start buying, making it too expensive to be affordable.
A very simple game, with a fairly short duration and a very low difficulty level for the author's standards but still fun and perfect as a game to be played on the beach with your “shades" on. There are not many locations to explore and the puzzles are never too difficult. With a little intuition you can easily figure them out and AT experts will be able to complete the adventure in a few hours. If this is the first time you've faced such a game, then "Lights of Finance" is perfect for getting into this magical and fascinating world.
So what are you waiting for? Download it and play it, you can do it from your mobile phone, just get an interpreter for AT Infocom. And at the end, do not forget to send an email with the password that will be revealed to you by the game. Send it to ovranilla@gmail.com: Marco Vallarino will rightly enter your name into the Trading Room, the hall of fame of "Luci della Finanza"!

by **Giorgio Balestrieri**





Courtesy by Elena Mascolo

OUR FINAL SCORE

» Gameplay 90%

A good vocabulary, a humorous plot and a low level of difficulty allow anyone to face and have fun with this title.

» Longevity ??%

It's a textual adventure and you know how it works with this kind of games, once it's over, it's done, you don't get to play it again, so talking about longevity really doesn't make sense.

WARNING: the game is available only in Italian language. If you are not familiar with Italian, it will be extremely hard to play this game.

WARDNER

Year: 1987
Developer: Taito

Platform: Arcade
Genre: Platform

In late 1980s, the platform genre was raging in the arcades and trying to emerge among so many legends such as Ghost'n'Goblins, Pacland, Wonderboy or Super Mario Bros. was certainly hard.
But in 1987 Taito released Wardner, a title produced by Taoplan that featured a fast, classic horizontal scrolling, worth of notice, and I think it deserved more luck and fame than it actually had.

In the game, we find ourselves in the ominous forest of Wardner when the young and chubby Pyros and Erika are taken aback by a magician sent by the terrible Wardner (it's really original that the villain os the one to give the title to the game for once!), who with a spell turns the young woman into a crystal ball and takes her away with him.

That's why we are forced to take a long journey to try to free her and defeat the evil Wardner in the final battle.
On our way we must wander through forests, abandoned villages and haunted castles.

The gameplay features two buttons, one for jumping and the other for shooting, which basically consists of a sacred fire that we can throw from our finger tip. During the game we can increase our firepower by collecting power-ups or items that protect us from the enemies, such as a cloak or even small elves. You can also buy all this stuff in a store that you find along the way and where you can spend all the coins collected during the journey.

Our hero has to deal with different kinds of monsters and at the same time he has to jump between platforms and avoid all kinds of obstacles. The sprites are very colourful and in perfect pixel art style, which always provides that magical touch to the arcade titles.

The rhythm is not as hectic as in many other games of this kind, on the contrary we must move carefully and evaluate every move or jump to stay safe or alive.
The audio and sound effects can't be hailed as a miracle but still they follow through the action and every scene very well.

In my opinion, one of the few weaknesses of this title is the difficulty of progressing into the higher levels. While traps and tricks are to be avoided and carefully studied, on the other hand, end-level bosses, including Wardner himself, are not entirely irresistible.

Still the title is very enjoyable to play and had quite a success at the time which led to the birth of conversions for Nintendo and Mega Drive.

by **Querino Ialongo**

OUR FINAL SCORE

» Gameplay 80%
Wardner's gameplay features nothing new for the genre. One button to jump and one to shoot, but the game is very fun all the same.

» Longevity 80%
The not too hectic pace compared to other titles might look like a limitation, in fact I think it was a careful designers' choice which makes this title pleasant even today, after more than thirty years.

The enjoyment of going against the tide

In the last two years the interest in our beloved hobby of retrocomputing has been growing strongly. The numbers speak for themselves. Old but still very useful tools like forums, "traditional" social networks (FB, Twitter, Instagram and YouTube) and younger ones (Discord, Twitch and Reddit) are witnessing a slow but steady growth of "permanent" fans as well as casual, nostalgic users. As it happens with any human activity, there are many types of users and as many approaches to the subject of retrocomputing. This passion we share provides so many features and several joining-on points for those who want to be part of the scene or contribute with concrete ideas and compelling projects.

You can start from purely collecting old machines, peripherals, programs and gadgets, and end up creating complete retrocomputing museums (full of working or inactive gear). You can even go from the unscrupulous accumulation of questionably valued hardware, up to the design of new homebrew machines in the style of old microcomputers. Some hobbyists like the bare and raw unboxing footage, some prefer intriguing retro-programming challenges. Some dive into their past by typing-in old BASIC listings taken from books and old magazines, some attempt the complicated refurbishing of obsolete systems. From the noblest initiatives aimed at preserving the history of computing to the lightest ones dealing with game streaming, there are countless other activities related to retrocomputing that we at RMW try to document and bring to the readers' attention with every release. Some are more attractive than others, some are more complex to deal with, but all are worthy of attention, because, as we well know, every time we talk about microcomputers, home computers, personal computers and 8 and 16 bit consoles, our ears stand up, thirsty as we are for modern novelties and rediscoveries linked to the past.

Recently I could witness, with a certain regret, the different feedback obtained from popular activities that are often arranged in Facebook retrocomputing groups in the form of online meetings and videoconferences. Only a few scattered spectators when it comes to the intimate functioning of computers on which we often spend a lot of time working, experimenting and playing. Many, so many participants when it comes to describing and showing arcade or 8/16-bit games of the past. Obviously no one expects a crowd of fans eager to know everything about CPUs, wiring diagrams, logic ports and ROM/RAM memories. That's just down to the intrinsic need for a minimum of technical background from viewers to deal with certain topics. Just as we're not surprised that when it comes to reaching a high-score on Pit Stop II or overcoming the R-Type end-of-level monster, even a small meeting can attract the attention of more than a hundred connected fans simultaneously.

Compared to 30 years ago, there are so many sources to learn, study and understand how things really work inside a computer (even when we run one of our favorite games), so much that we would expect a greater interest in disciplines undeniably more complex but certainly able to give back great enjoyment. Unfortunately the "sign of the times" is certainly not reassuring in this respect. Only a few people wish to approach with passion and dedication the study and understanding of the systems we loved so much in our youth.

As a retrocomputing magazine, we will always promote everything that has "retrocomputing" written on it. It is our own mission, after all. Still it is necessary, in our opinion, to go a bit against the tide, so that a whole baggage of notions, principles and practices (such as the construction and reproduction of electronic circuits, low-level programming, software techniques that manage to overcome the limits imposed by hardware, a deep understanding of the mechanisms that regulate the operation of our beloved machines, whether in physical version, emulated or implemented in FPGA), will not be lost or remain buried in some obscure books of the past. So, dear readers, that's an invitation to you all. Try to get into one of these topics and you will see that they are not that hard to approach. We are still in time. Take pen and paper (or your keyboard and wordprocessor) and tell us about your "hardcore" or "hard-coded" experiences!

David La Monaca

RetroMagazine World
Year 1 - Issue 2 - AUGUST 2020

Chief Editor
Francesco Fiorentini
Managing Editor
David La Monaca
Editing Manager
Marco Pistorio
Web Manager
Giorgio Balestrieri